Perspectives on Distributed Computing: November 2007 Quarterly Report

Lisa Childers (childers@mcs.anl.gov)
Lee Liming (liming@mcs.anl.gov)

Computation Institute
Argonne National Laboratory
University of Chicago
December 14, 2007

Report Contents

1 User Perspectives Project Overview

The User Perspectives team has been interviewing individual members of collaborative science projects who use distributed computing technology. As part of the Community Driven Improvement of Globus Software (CDIGS), the User Perspectives project is documenting the work-related goals, methods, and challenges facing today's scientific technology users and is recording their perspectives on Globus software and its relevance to their work.

The primary deliverable of the project will be a final paper describing the findings from the interviews (see Fig. 1). The project is generating a number of immediate benefits along the way. This interim report, *Perspectives on Distributed Computing: November 2007 Quarterly Report*, provides an early look at the project's accomplishments.

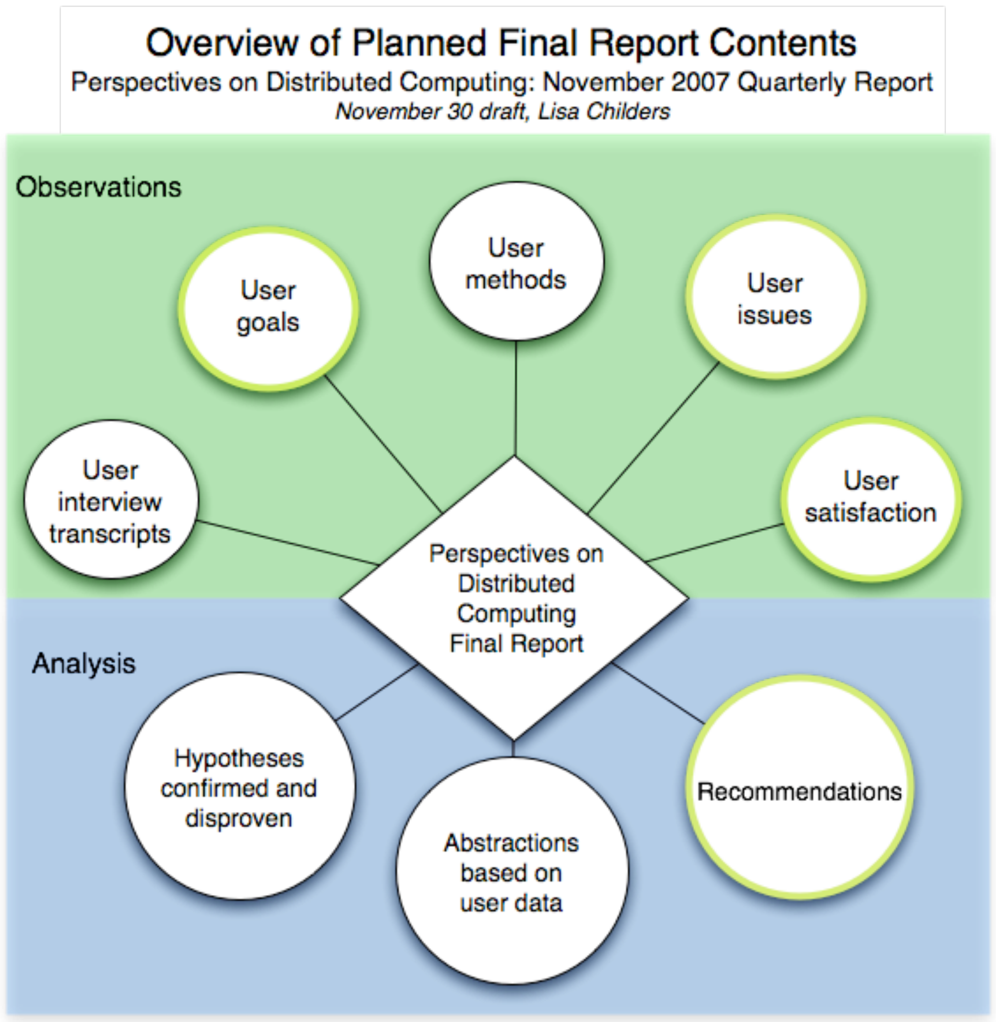

Overview of Planned Final Report Contents
Perspectives on Distributed Computing: November 2007 Quarterly Report
November 30 draft, Lisa Childers
Observations
User goals
User methods
User issues
User interview transcripts
User satisfaction
Perspectives on Distributed Computing Final Report
Analysis
Hypotheses confirmed and disproven
Abstractions based on user data
Recommendations

Figure 1: Planned final report contents for the User Perspectives project

At this early stage, the report includes preliminary summaries of user goals, issues, and expressions of user satisfaction extracted from the interviews transcribed to date. The interviews themselves contain information on additional topics, so an increasingly rich set of observations and analytical material will appear in subsequent reports. In the final report, interview transcripts will also be included in their entirety. Although we hope the reader will find this quarterly report useful, its preliminary nature should be kept in mind.

The User Perspectives project is the cornerstone of the CDIGS User Advocacy program. An overview of that program, including a brief description of the work areas and recent accomplishments, can be found in Appendix A.

2 Summary of Quarterly Progress

During this quarter, users were interviewed from a variety of U.S. projects spanning several scientific fields. Interviews were transcribed, and observational summaries were created. A high-level description of the users contributing to the observations was produced, as well as summaries of user goals, issues, and expressions of satisfaction. The report also presents our first set of interim recommendations.

2.1 Users Interviewed

From August 2 through October 16, 2007, eighteen interviews were conducted with users of scientific distributed computing technology. The individuals interviewed represent seventeen projects spanning twenty-one scientific fields. Nine funding agencies sponsored the projects. Each project is based in the United States, though several include international partnerships.

During the approximately hour-long conversations, the users talked about their work-related goals and challenges. Project affiliations of those interviewed this quarter included Access Grid, Adaptive Cyberinfrastructure for Threat Management in Urban Water Distribution Systems, Angle, CEDPS, FermiGrid, GNARE, GridShib, MPICH-G2, nanoHUB, Network for Computation Nanotechnology, Optiputer, PASTA, PUMA2, TeraDRE, TIGRE, UCLA Grid Portal, and the VO Services Project.

When asked to identify their job type, the eighteen users described themselves variously as application engineer for scientific computing, assistant group leader, computer professional, developer, lead scientist, middleware architect, programmer, programmer analyst, principal investigator, professor, project lead, research programmer, scientist, system administrator, system architect, and technical director.

The respondents were also asked to name the scientific field(s) associated with their project. Answers included astronomy, bioinformatics, biology, bioscience, civil engineering, collaboration technologies, computer science, cosmology, ecoinformatics, education and outreach, geoscience, genomics, Grid security middleware, high energy physics, hydrology, information technology, meteorology, nanotechnology, non-accelerator physics, non-high energy physics, and visualization.

The funding agencies supporting the respondents' work included Brookhaven National Laboratory, the Department of Energy Office of Science, Long Term Ecological Network Office, Microsoft Research, the National Science Foundation, the State of Texas, and the University of California Office of the President.

2.2 Transcriptions Completed

Ten interviews were transcribed during the reporting period, totaling approximately 42,000 words and eleven hours of interview time.

2.3 Observations Summarized

Last quarter we generated the first set of observational summaries, categorized by "high-level descriptions of users," "issues raised by the users," and "expressions of satisfaction." This quarter we updated those summaries to include the new data from this reporting period. We also added a new summary category: "user goals."

The updated high-level descriptions of users show that Globus software is being used in a wide set of environments. This result continues to support the notion that the requirements addressed by Globus software are fundamental to problems in many scientific fields.

The issues summary continues to include both issues with specific Globus components and issues with using Globus (and other) software in production environments. A significant number of the issues reported concern collaboration and other social matters.

The user satisfaction summary identifies specific aspects of Globus components that users find valuable.

The new summary category, user goals (Section 3.2), helps answer the question, "What kinds of science does Globus support?" The diversity of scientific goals is both gratifying and humbling. We note that most, but not all, of the interviewees are currently using Globus software.

Another new category of observations—user methods—was initiated during the reporting period but not completed. Preliminary results are promising in terms of providing an interesting perspective of the distributed computing user. Top-level classes currently identify methods for "interacting with people," "interacting with technology," "conducting science," "providing infrastructure," and "developing products." An initial methods summary will appear in the next quarterly report.

2.4 Issues Dispatched

User interviews often point to specific issues that individuals or even many users are experiencing in the field. While the greatest benefits of the User Perspectives project will be realized with the documentation and analysis of dozens of user interviews, issues that might warrant immediate follow-up are dispatched to appropriate CDIGS team leaders for further investigation. This section summarizes the issues dispatched for the reporting period.

2.4.1 Number Dispatched by User Project

- 1 issue from Lattice QCD
- 1 issue from MPICH-G2

2.4.2 Number Dispatched Relating to Globus Technology

- 1 issue for Swift
- 1 issue for RLS/DRS
- 1 issue for GridFTP/RFT
- 1 issue for Service Hosting

2.4.3 Number Dispatched to Globus Project Representatives

- 1 issue for TeraGrid

2.5 Study Protocol Approved

During the reporting period the User Perspectives study protocol was reviewed and approved by the University of Chicago Social and Behavioral Sciences Institutional Review Board.

3 Highlights of the Observations

This section begins with a high-level description of the users contributing to the study, followed by three types of observation: descriptions of user goals, reports of user issues, and expressions of user satisfaction. Observational excerpts have been edited for flow and readability.

3.1 *User Overview*

The observations in this report are based on transcripts produced to date. Figure 2 shows an overview of these users. A high-level description of each follows the figure.

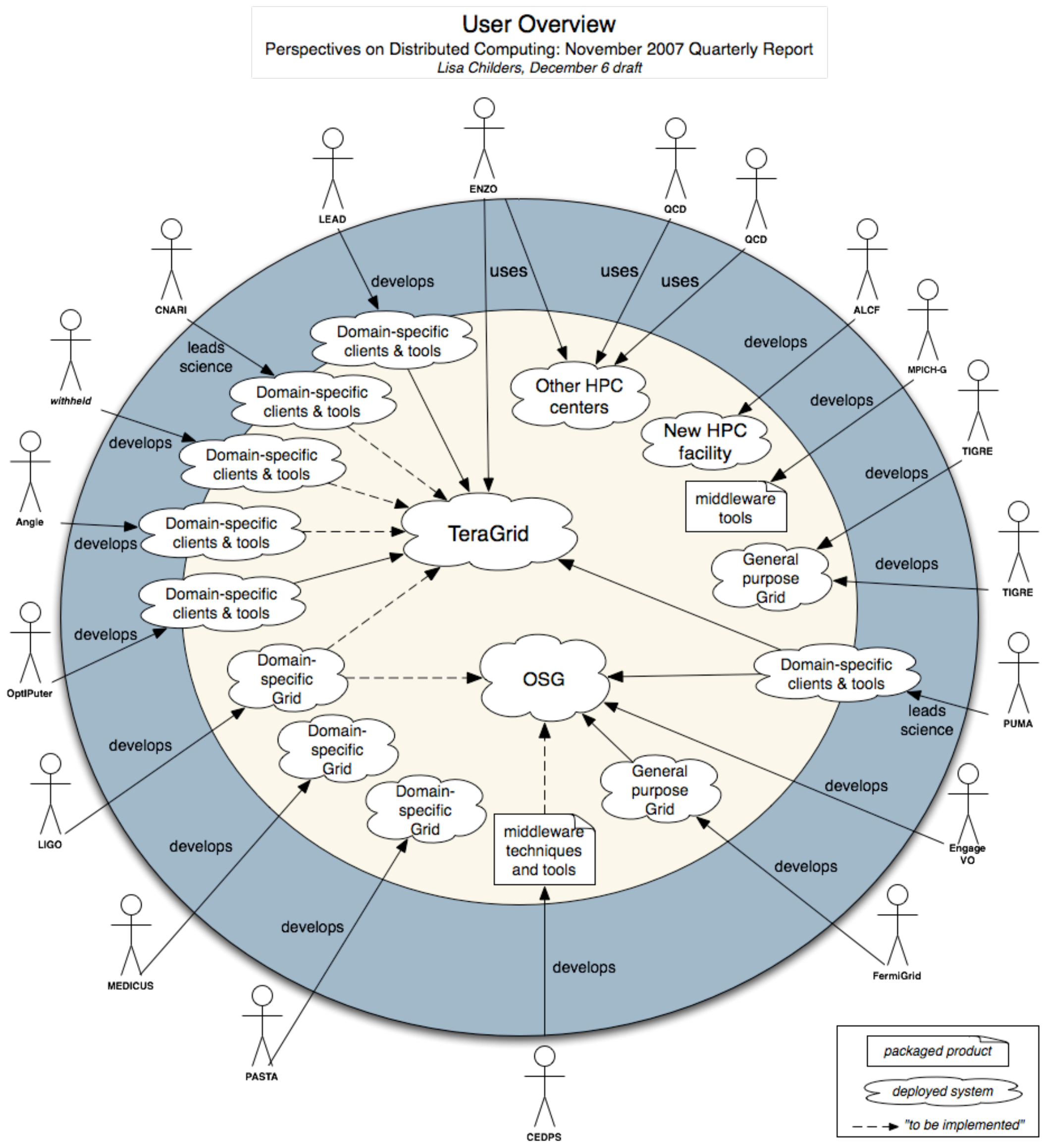


Figure 2: Overview of users covered by this report

3.1.1 System Administrator/Research Programmer

September 2007 interview
Featured interview quote: "**The end goal is to automatically detect network anomalies**."
User27 develops infrastructure enabling the identification of anomalous network behavior. *User27* supports domain scientists – developers of data mining algorithms – by providing easy and efficient access to the data. Currently the infrastructure is running on local a local cluster, but plans include offloading processing to distributed computers. *User27* provides a system developer's view of today's concerns in porting application code to the Grid.

3.1.2 Project Lead

September 2007 interview
Featured interview quote: "**Our goal is to make it easier to troubleshoot Grid applications**."
User26 creates techniques and tools to make it easier to troubleshoot Grid middleware applications. The current focus of the project is working with middleware vendors to adopt a common logging format. Future work includes creating a central logging facility and analysis of the data. *User26* not only discusses methods for troubleshooting distributed systems, but also for interacting with potential adopters of technology.

3.1.3 Lead Scientist/System Architect/System Administrator/Developer

September 2007 interview
Featured interview quote: "**We work to enable discovery, access and synthesis of distributed datasets**."
User21 is building a central data warehouse for storage and synthesis of ecological data. The data are drawn from over twenty sites distributed across the U.S., Antarctica, Tahiti, and Puerto Rico. Future use scenarios of the system include enabling access to distributed computation in order to run simulations on the warehoused data. An overriding goal is to broaden the scope of the science conducted, which is currently at site scale but will be moving to national scale and perhaps global scale. Other project foci include the topics of metadata, provenance, and security.

3.1.4 Project Lead

September 2007 interview
Featured interview quote: "**We can provide our users with fresh data more frequently because of the Grid**."
User20 is the domain expert driving development of an application that enables biologists to analyze large volumes of genomic data. To facilitate analyses, *User20's* team warehouses genomes from over twenty biological databases. The application under development integrates these data with a suite of domain-specific tools and computational resources, providing biologists with an interactive system for high-throughput analysis of genomes and metabolic reconstructions. *User20* and his team are a good example of how distributed computing can help science to scale.

3.1.5 Computer Professional/Assistant Group Leader

September 2007 interview
Featured interview quote: "**GRAM2 is kept alive by the need to interoperate with European experiments**."
User17 develops and maintains a general-purpose institutional Grid to pool and share system resources and to unify several system-level services. *User17* provides a seasoned,

practical perspective of several key issues facing today's middleware consumers – from both the client side and the server side. Key topics for *User17* include how to deal with technology failures, manage deployments comprising thousands of nodes, and maintain interoperability with both domestic and international Grids.

3.1.6 Project Lead

September 2007 interview
Featured interview quote: "**We assume a world where lightpaths can be scheduled between computers**."
User16 works on an experimental systems project designed to understand how distributed system architectures might look in an environment where bandwidth is unlimited. Working with TeraGrid infrastructure, project members use tens of gigabits of bandwidth in their experiments. *User16* outlines an approach for both investigating and evaluating experimental architectures.

3.1.7 Professor of Computer Science/Principal Investigator

August 2007 interview
Featured interview quote: "**I start with benchmarks and follow up with real applications**."
User14 develops a middleware parallel computing tool specifically designed to help solve problems that do not fit on a single computational resource. This work leverages the services and libraries that already exist in the distributed computing ecosystem. As such, *User14* describes a method and associated challenges in working with both domain experts and middleware providers.

3.1.8 Scientist

August 2007 interview
Featured interview quote: "**We play a strong bridge role in connecting people with technology**."
User13 builds general-purpose Grid infrastructure in support of biosciences and medicine, energy exploration, and air quality modeling for the state of Texas. A significant portion of *User13's* time is spent working with members of the statewide scientific community, helping to bring their applications to the Grid. In addition to insights related to community engagement, *User13* outlines some current issues in security deployment.

3.1.9 Scientist

July 2007 interview
Featured interview quote: "**The right approach is to be highly collaborative with domain specialists**."
User12 is a domain expert whose team uses database technology to store neurophysiological data for use by scientists. The project team includes Grid experts, who work to leverage computational resources for processing data. *User12's* application area involves research in the recovery from stroke.

3.1.10 System Designer/Developer

July 2007 interview
Featured interview quote: "**Resource usage within our Virtual Organization is opportunistic**."
User11 builds infrastructure to lower the barrier to entry for the Open Science Grid; a complementary goal is to bring new users onto OSG. *User11's* project does not own any

OSG resources: it provides users access to resources left unused by the major OSG-based experiments. Much of *User11's* work involves detecting and fixing problems with OSG infrastructure before users encounter them. *User11* provides a glimpse of both the challenges and opportunities afforded by today's distributed computing tools.

3.1.11 Storage Engineer/Project Lead

July 2007 interview
Featured interview quote: "**Solving problems is easy once you have all the data**."
User10 is building the storage and I/O systems for a new leadership-class computing facility, due to come online in late 2007. Many of *User10's* future users run their applications at HPC centers that exist today. The new facility will offer scientists opportunities to scale their work by increasing the size of a mesh or the number of molecules simulated, for example. *User10* describes issues associated with providing data services that will operate at unprecedented scale.

3.1.12 System Architect/Scientist

July 2007 interview
Featured interview quote: "**Globus enables more science**."
User9 builds infrastructure for distributing gravitational wave data detected from astrophysical sources to analysts around the world. Having created a data replication Grid that is robust and reliable, the *User9* team is now working to reduce the time needed to configure, maintain, and manage this Grid. Other work includes enabling production data analyses to run across a greater number of resources, with plans to federate into Grids such as OSG or TeraGrid. *User9* provides a view on the challenges scientists face keeping track of data: not only coming off the sensors, but also tracking results from Grid-facilitated analyses.

3.1.13 Professor

June 2007 interview
Featured interview quote: "**I am trying to understand where Grid computing adds value**."
User8 works in the area of lattice QCD, a numerical approach to quantum field theory. *User8* runs teraflop year-scale calculations, producing hundreds of files, which must be archived for use as input for later job runs. Curious about whether Globus could help with data movement, *User8* shares valuable insights regarding the challenges of using Globus as an HPC facility user.

3.1.14 System Architect

June 2007 interview
Featured interview quote: "**If things don't work, you need an expert to fix them**."
User7 is building a Grid enabling the state of Texas to provide scientific applications for research and education. A secondary goal of the effort is to help develop the business of the state, in terms of providing greater access to resources and knowledge. *User7's* team provides high-level support, helping users translate their applications to the Grid. *User7* describes the team's approach for building the Grid and outlines some of the problems infrastructure providers face in building these systems.

3.1.15 Scientist/Developer/Project Lead

June 2007 interview

Featured interview quote: "**The Grid idea is great, but there are barriers to making it work today**."

User6 is an experienced lattice gauge theorist working to understand the interactions of quarks and gluons and applying that understanding to the discovery of new, fundamental parameters of elementary particles. Looking at Globus for help in managing lattice files, *User6* provides important feedback on the barriers facing new users.

3.1.16 Developer/Scientist

June 2007 interview

Featured interview quote: "**Performance improved from days to seconds**."

User5 is in the fourth year of building infrastructure enabling radiologists in North America to share medical data in research and clinical settings. The *User5* team is creating higher-level capabilities by integrating domain-specific code with the programmatic interfaces of Globus services. The project is one of the few open source medical applications in existence today, and its successes are serving as a catalyst for change in the field of medical technology.

3.1.17 Developer/Scientist

June 2007 interview

Featured interview quote: "**The reason my tasks are so time-consuming is failure**."

User4 maintains a simulation code that runs at the largest computing scale currently available. The focus is on the HPC aspects of the simulation as opposed to the science: fixing algorithmic deficiencies to make the code run at the next largest scale. The application is demanding, not only in terms of CPU usage, but also in terms of producing prodigious amounts of data. Because the application pushes so many system boundaries, *User4* encounters problems not experienced by the "thousands of users chipping away at small jobs."

3.1.18 Scientist/Developer

June 2007 interview

Featured interview quote: "**The Grid is a black box to me**."

User3 is building a framework that enables neuroscientists to store, analyze, and share data. The user is not a Grid computing expert but collaborates with people who are. He depends on these experts to translate his domain-specific needs to the Grid. The group is working toward using TeraGrid resources to scale processing of high-volume domain-specific data.

3.1.19 Science and Technology Liaison/Portal Developer

May 2007 interview

Featured interview quote: "**Troubleshooting currently requires knowledge about software internals**/"

User2 works with atmospheric and computer scientists to implement meteorological use cases in a portal environment. *User2* simultaneously assumes the role of a CS user and scientific application developer, working on integration and testing of distributed computing tools and clients. The Globus services used are maintained by the TeraGrid. Familiar with Grid computing concepts, *User2* provides a seasoned, practical view from the client side.

3.2 *Highlights of User Goals*

Figure 3 shows an overview of the user goals observed in the study to date. In each of the following subsections, individual observations are captured in the bulleted items.

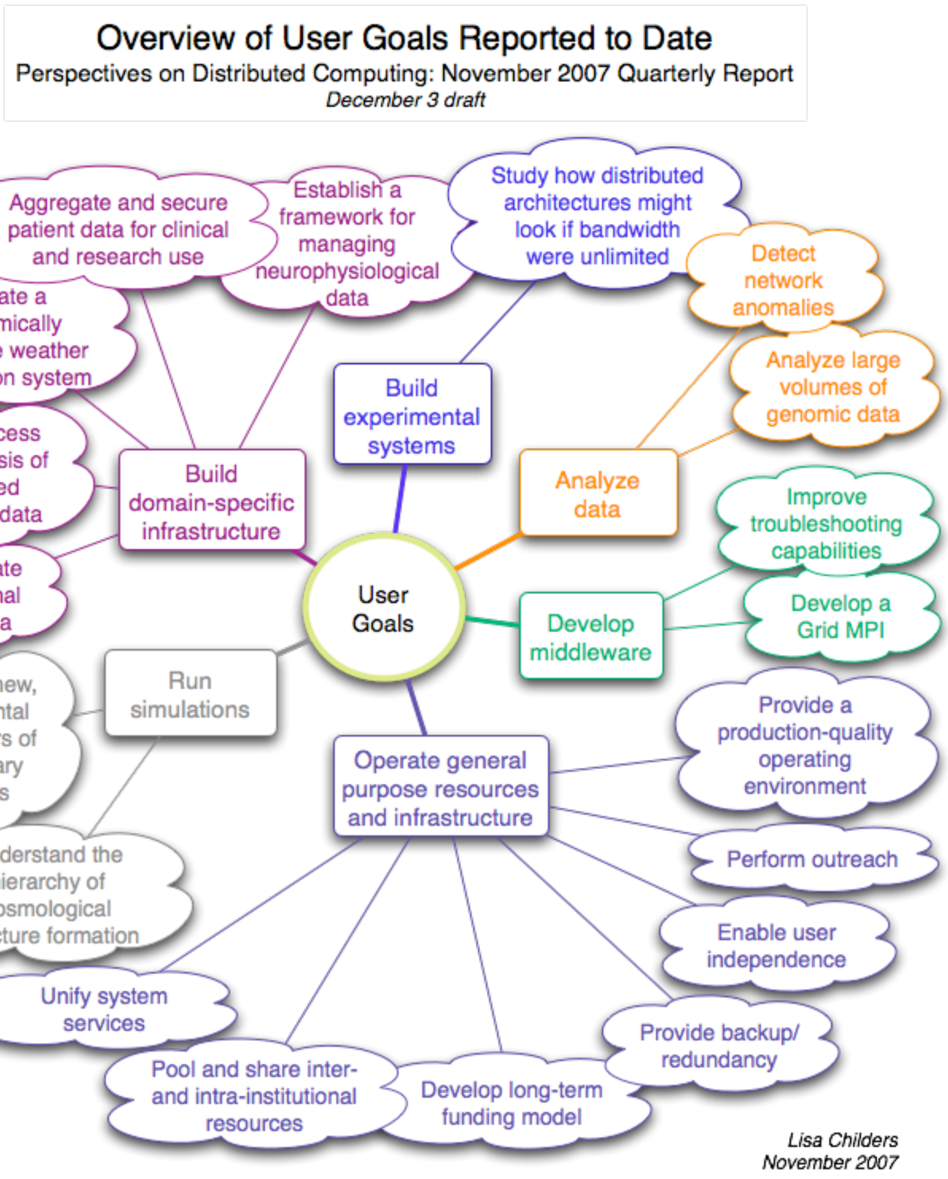

Overview of User Goals Reported to Date
Perspectives on Distributed Computing: November 2007 Quarterly Report
December 3 draft
User Goals
Build domain-specific infrastructure
Aggregate and secure patient data for clinical and research use
Establish a framework for managing neurophysiological data
Create a dynamically adaptive weather simulation system
Enable access and synthesis of distributed ecological data
Disseminate gravitational wave data
Build experimental systems
Study how distributed architectures might look if bandwidth were unlimited
Analyze data
Detect network anomalies
Analyze large volumes of genomic data
Develop middleware
Improve troubleshooting capabilities
Develop a Grid MPI
Run simulations
Discover new, fundamental parameters of elementary particles
Understand the hierarchy of cosmological structure formation
Operate general purpose resources and infrastructure
Provide a production-quality operating environment
Perform outreach
Enable user independence
Provide backup/ redundancy
Develop long-term funding model
Pool and share inter- and intra-institutional resources
Unify system services
Lisa Childers
November 2007

Figure 3 User goals reported to date

3.2.1 Building Domain-Specific Infrastructure

In user interviews conducted thus far, goals related to domain-specific infrastructure fall into five categories. The first category involves **building infrastructure for gravitational wave data**:

- The main goal of the project is to detect gravitational waves and to conduct gravitational wave astronomy – in other words, to learn about our universe through the use of gravitational waves. Diving down a bit, my real project is to develop the project's DataGrid. The purpose of the DataGrid is to enable as much science as possible to be conducted using the project data. The data has to be analyzed. It's very computation and data intensive. And the main goal of my project is to build infrastructure (tools, middleware, end-user tools, services, and systems) that enables scientists to efficiently analyze the data and conduct their research.

- The focus of the DataGrid is first of all to make data available after it is saved – comes off the instruments – make it available to all the analysis sites. This enables scientists to efficiently use the data sites around the world to analyze project data.

- The second focus of my work is on the data analysis side, where I'm trying to get some of these production workflows to run across the DataGrid. I'm working on getting some of the data analysis pipelines or workflows to run in more sophisticated ways across multiple computing sites. Right now our users tend to pick a site and go run there. And while they use some Grid tools to do some of the data finding, they don't typically use Grid tools to leverage more resources than are available at a single site. So another of my metrics for the coming year is to try to enable production data analyses that run across multiple DataGrid sites in a continuous way.

- I'd like to try to get production data analysis running on sites that are external to the DataGrid. So federate into other Grids, such as Open Science Grid and TeraGrid.

- Moving forward, we want to build WSRF-compliant services on top of both Globus Toolkit 4.0 Java and C WS Core.

- We plan to put up really nice dashboards that our collaborators can look at and see the current state of replication throughout our DataGrid.

- The goal I'm really looking for is to have a mean time between failures of three months. That's what I'm targeting from the data replication side.

The second category is related to **building infrastructure for weather simulation**:

- The science goal is building a dynamically adaptive weather simulation system. The engineering goal is to build a service-oriented architecture in support of the science goal. So we are building the service-oriented architecture and Grid middleware ourselves while trying to leverage as much as possible the tools already available in the community.

- What we are trying to do is integrate computation systems with atmospheric science models and instruments. From the science perspective, we are trying to look for more ways of running the huge computational science models at high resolutions. That's been a big challenge. So running a weather forecast at storm scale and at tornado scale is something we've been trying to do.

- From the computer science part, we've been trying to make all these legacy Fortran codes and the legacy applications run in a service-oriented architecture and providing users direct access to advanced computational resources from a portal-based environment, so users don't need to learn about how to use a particular cluster or job manager.

The third category of domain-specific infrastructure goals is related to **building infrastructure for neuroscience**:

- We're establishing a framework where neuroscientists, especially people who are involved in brain imaging, can store, analyze, and share their data in an effective manner.

- Our project is aimed at using database technology to store data that is electrophysiological or neurophysiological in nature. The data consist of hundreds or thousands of timepoints in a time series. Each timepoint consists of a large amount of data itself, such as a brain image or an electrophysiological recording from different sites on the brain. Using database technology to store the data is a way of allowing much more useful access to the data. Also through interactions with Grid computing experts at the University of Chicago, we found that it's a much better way to interface with distributed and high-powered computing devices in order to process the data. This is all done in the context of research in the recovery from stroke, which is a disease of brain vessels that is very devastating.

The fourth category is related to **aggregating and securing medical images**:

- The main goals are to efficiently and compliantly communicate medical images in a secure fashion so that patient privacy is guaranteed, using existing security mechanisms and other standards-based technology provided by the Globus Toolkit.

  An example of the vision is that a patient comes into the hospital, and this patient's record is not only existent in the hospital, but also available on the Grid so that other healthcare providers can access the data and also add to it, so that wherever you go as a patient, the Grid can basically follow you, aggregating all the information that exists for you at various health providers, and collecting them at the point of care. The challenge is to provide a technical solution that can scale to allow a large number of healthcare providers to interact and share images.

- We also work on the security model to make sure the privacy protection is there. This will actually be very important to further explore the medical field. When you have a patient-doctor relationship, you as the patient sign consent that only the doctor who is treating you or the staff of that facility is allowed to see your medical charts. So now Grid enables us to communicate all these medical information wherever we want.

- One motivating factor is to develop a security model that allows the same doctor-patient relationship being translated onto the Grid, allowing data access, but only if the patient authorizes it. So we are developing what we call a patient-centric authorization model as a way to approach this. What we want to accomplish in the project is to show that there's technology available today, such as the very strong security infrastructure in the Globus Toolkit, which can be used in an intelligent way to build a security model for patient authorization and privacy for health data.

- Specific goals within that project are to ensure that the radiology imaging workflow is flowing. That means that there are very different modalities out there that we must take care of, and there's a lot of incompatibilities between the very different devices. Because of that, one current work focus is to make sure that the project can handle all these different vendor-specific imaging devices.

- Now that you have all these images collected, what are the cool questions? Some cool questions, if you have many, many deployments are, “Give me all the computer tomography images of the twenty-year old males who have lung cancer.” These are totally relevant questions to ask in the medical domain, but we cannot do that today because the data are not aggregated. Look at the National Institutes of Health, which is a major medical research sponsor in this country: if they accumulate 10,000 cases per year in their archives, that would be a lot for them. If you go to the radiology department of a small neighborhood hospital, they probably do around 60,000–100,000 cases per year. Then you have, for instance, probably 80–300 hospitals in Los Angeles County. You get the idea that in the clinical domain on a daily basis you can get way more data then you can ever get in research. The critical thing here is that if you build a Grid which is connected to clinical data you can come up with very interesting epidemiological questions. “Give me all the cases of a specific disease in a specific area of the country” –and then you can see a big picture.

The fifth domain-specific infrastructure category relates to **building a warehouse for ecological data:**

- The primary project involves an architecture named the Provenance Aware Synthesis Tracking Architecture. As part of our research network there are 26 sites, which are spatially distributed across the continental United States, two in Antarctica, one in Tahiti, and one in Puerto Rico. Each of the sites is collecting scientific data. The goal of the architecture is to pull data from them in a seamless way, based on both the metadata records and open access to the actual data file. These data are brought into a centralized data warehouse to enable data discovery, data access, and synthesis of distributed datasets within the network.

- Once the data are actually centralized, we develop interfaces to enable users to explore the data. I think one term is “exploratory.” So we’re developing Web-based applications that include discovery interfaces, plotting routines, different types of data download mechanisms, and allowing end users to integrate different datasets so they can generate more or less synthetic products on the fly.

- Part of our goal is to make this next leap from science that takes place at the site, to science that takes place at the research network level. This would be a national, if not global, scale. So the anticipation is, once these datasets become available to end users, that simulation and modeling will begin taking place. That's probably on the horizon within the next 1–5 years.

3.2.2 Running Simulations

Two types of simulation-related goals have been observed in user interviews thus far, the first is related to **understanding cosmological structure formation**:

- As far as the cosmological part of it goes, we're trying to understand the hierarchy of structure formation on various scales, from the larger scales and the universe, down to galactic scales. This is a tremendous number of orders of magnitude in physical scale – in space and time.

- We hope to be able to account for observations, to determine the cosmological parameters of the universe we're living in. To provide theoretical underpinnings for observations from things like the Hubble space telescope, or the James Webb space telescope, or any of those major projects.

- The big challenge for us is in three-dimensional radiative transfer, which is how light basically interacts with a fluid medium. Our code is being extended right now to incorporate these frontier pieces of physics, that up to now we haven't had enough computer power to include. So, there is a finite list of things we'd want to add. I wouldn't want to be too strict about this point, but we'll probably add most of the physical things we want to add in the next two to three years. And then it will be mainly a question of just how much computer power can we get our hands on. Our goal will be to be able to run problems that continue to match the most powerful resources available in the U.S., whether they're DOE or NSF.

The second type of goal in the simulation category involves **elementary particle simulations**:

- The main goals are to understand quantum chromodynamics (QCD), which is one component of the Standard Model of elementary particle physics. This includes
  - calculating the masses of particles that interact strongly; those particles are made out of quarks and gluons,
  - calculating their decay properties, and
  - studying QCD at very high temperatures, and possibly with nonzero chemical potential.

- The goals are to understand the interactions of quarks and gluons and apply that understanding to the discovery of new, fundamental parameters of elementary particles.

3.2.3 Operating General-Purpose Resources and Infrastructure

In user interviews conducted thus far, goals related to operating general-purpose resources and infrastructure fall into seven categories. The first involves **providing production-quality operating environments:**

- Our goal is to provide stable computing facilities for people to do cutting-edge science. We are not about doing science; we are a resource/infrastructure provider. So our primary goal is to keep the equipment up and running, have as stable an environment (i.e., not changing, as few crashes as possible.) So in the spectrum between complete research software, which compiles once and never runs again, and bulletproof production software that's commercially available, we're closer to the production end of the spectrum.

- The project was composed as a demonstration project in the sense that we have to demonstrate the capabilities in these areas. But we're just now transitioning into creating the conditions for a production-scale Grid.

- The project really has as its goal producing production-quality infrastructure that will take less effort to maintain after it was created.

Many of the operators of general-purpose infrastructure pursue **outreach-related goals**:

- We help new users get on to OSG - especially users that are not in high-energy physics. The idea is to demonstrate OSG infrastructure.

- The whole idea is that we approach users who have smaller projects and smaller teams. They usually have very little IT or computer science knowledge. They have knowledge enough to write the model or to implement their idea, but that's pretty much where things end.

- We're charged with bringing up Grid applications in three targeted application areas: biosciences and medicine, energy exploration, and air quality modeling. These areas were chosen as examples of the application of Grid technologies to economically useful and interesting topics.

- Our main goals are to demonstrate applicability of Grid technologies to a wide variety of economically interesting and intellectually useful activities in the state.

- We try to bring in industrial partners and to help the business of the state as well, in terms of access to more resources and access to new knowledge and collaborations.

- We seek to support science in research and education around the state.

- Our basic goal is to take Grid technology deeper into the academic infrastructure than it has gone so far. Our particular project is targeted at the state of Texas, but we're also working with the Open Science Grid, SURA (the Southeastern Universities Research Association), and several regional organizations.

One infrastructure maintainer cited **enabling user independence**:

- After a couple of weeks, the goal is for them is to be able to do their own runs without emailing us questions.

Another mentioned the need to **develop a long-term funding model**:

- The main goals of the initial phase are to
    - have an operational system
    - have an initial set of users - scientists, researchers and educators - using it
    - to begin to form the collaborations and relationships to keep it going longer term

Two goals are related to **enabling redundancy/back-up facilities**:

- One of the motivators for this data collection effort came out of a weather event. A collection in Houston was down for several days due to a hurricane, because they turned the machines off and put them in a truck to move them somewhere else. So part of the goal is to be able to replicate data around the state so that kind of thing won't happen.

- Our main focus for this year is making it all redundant, making the infrastructure all highly redundant so we don’t have any one single point of failure.

Pooling and sharing of resources is also cited as a motivator:

- A goal is to create a structure within our site so that six or seven interest groups with big pots of computing can share each other's resources. This is by far the bigger focus: to share the resources amongst ourselves, as opposed to sharing resources with people outside our site.

- One goal is to provide a unified point of access for inbound Grid jobs, in order to share site resources with the Open Science Grid.

Unification of system services was observed as a seventh goal type of the general-purpose infrastructure operators interviewed to date:

- Another important aspect of our Grid is that it provides a unified authentication and authorization structure for site resources. There are currently seven local clusters included in our Grid, with upwards of several thousand batch slots available at any given time.

- One goal is to expose mass storage in a shared way.
- One goal is to have a unified systems support structure.

3.2.4 Developing Middleware

Two middleware development goals have been expressed in user interviews thus far: **improving troubleshooting capabilities** and **developing a Grid message passing interface (MPI).**

- The goal is to make it easier to troubleshoot Grid middleware and Grid applications, where troubleshooting doesn't just include failures but also includes performance-related issues. The nature of Grids makes it quite difficult to figure out the source of failures, and much of the underlying middleware lacks the right hooks to make it easy. So the goal of this research project is to figure out what is missing and try to get it added. We are trying to get many pieces of Grid middleware to use a common logging format and to log the right stuff. The hope is for OSG to report a noticeable drop in number of failed jobs and also to report a decrease in the amount of time it takes to track down problems.
- We are creating a tool, which is a Grid MPI, that can be used to solve computational problems that cannot otherwise be solved. Every time that we have solved a problem that no one was able to do before, we are very happy because our tool is enabling technology. That's our goal. And we keep pushing that envelope farther and farther out – as far as we can. I also have a personal interest in trying to make this stuff work as a way of improving usability.

3.2.5 Building Experimental Systems

One experimental system-related goal has been observed in user interviews thus far: **studying how distributed architectures might look if bandwidth were unlimited**:

- The focus of our project is to study what happens to distributed computing in an environment where there is no bandwidth limitation. It assumes a world in which people can schedule dedicated lightpaths between distributed computers, on demand, in the same way that they would schedule supercomputer clusters. We want to understand how those assumptions change application architectures, as well as change application users' perceptions of high-speed networking. The overarching goal is to identify the bottlenecks in trying to take advantage of high-speed networking:
    - What are the middleware capabilities that need to be developed that are still missing?
    - What will the endpoints look like that connect to these high performance services? Will they be desktop computers? Will they be browsers?

3.2.6 Analyzing Data

Two data analysis goals have been expressed in user interviews thus far: **analyzing large volumes of genomic data** and **detecting network anomalies**.

- The main goal of the project is the analysis of large volumes of genomic data. When genomes are sequenced, they are represented as strings of letters, which signify different nucleotides. Once a genome is sequenced you want to know what functions the genes perform and what physiological and metabolic processes the genes are involved in. So, starting from just the alphabet soup of the sequence, by the end of the analysis you know
  - how many genes this organism has,
  - what they do,
  - how this organism lives (because we're reconstructing double helix properties),
  - whether it has any pathogenic or non-pathogenic factors,
  - what it transports in the cell, and
  - what it produces.

  So pretty much by the end of the analysis, not through experiments but by using pure bioinformatic methods, the biologists know quite a bit about the organism already.

- The end goal is to automatically detect network anomalies. As a first step toward that goal, we want to be able to interactively analyze the data to gain better understanding of it. This will allow us to devise better algorithms that will automatically do that for us. Such anomalies could include a user transferring large amounts of data, the presence of a probe, or some kind of an attack – any kind of anomaly, but we're looking at the problem from a behavioral point of view, as opposed to mining actual content.

3.3 *Highlights of User Issues*

User issues spanned a wide range, from machine-related (e.g., reliability) to communication to technology issues. Figure 4 shows an overview of these issues observed to date. Individual observations follow the figure.

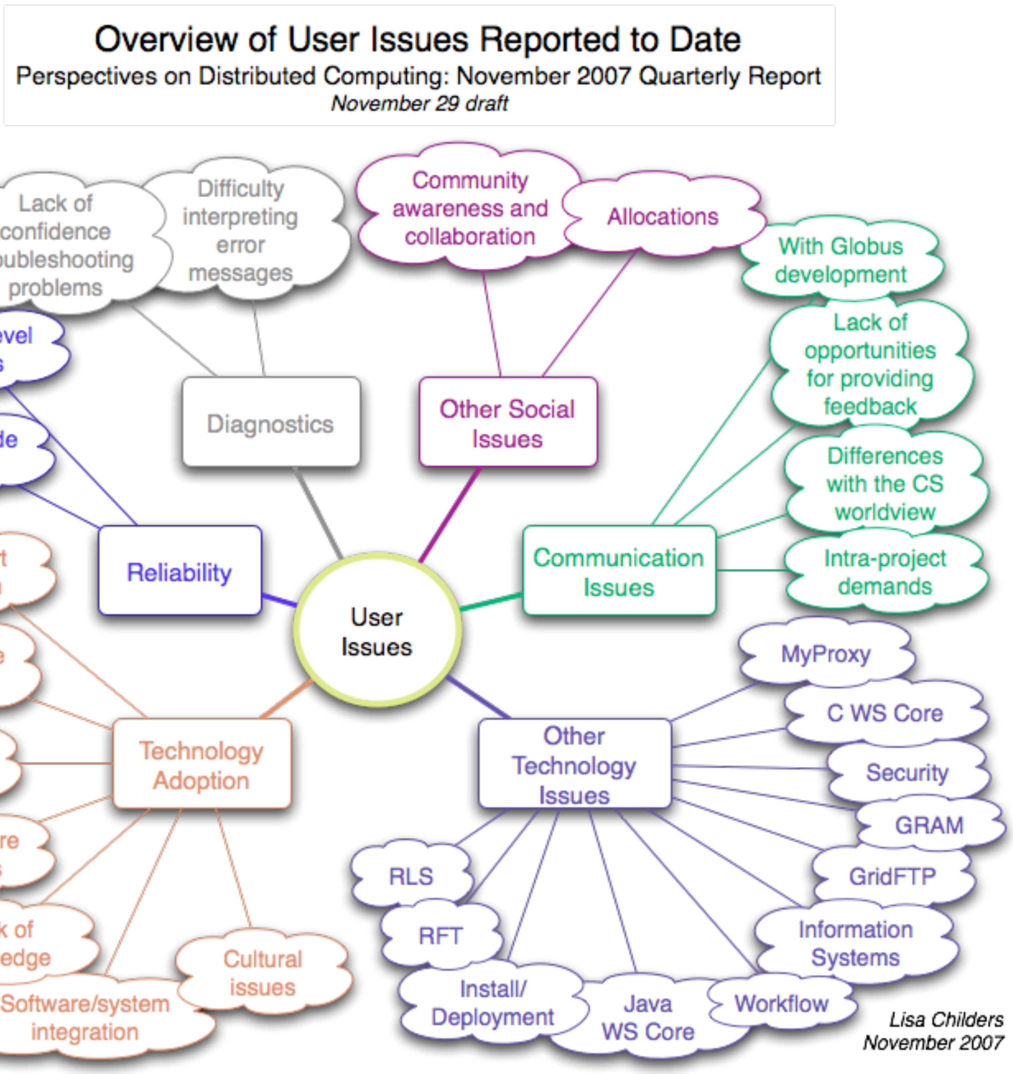


Figure 4: User issues reported to date

3.3.1 Reliability Issues

Two types of reliability concerns have been expressed in user interviews thus far, both of which suggest that additional fault tolerance and resiliency improvements would be well received.

3.3.1.1 Systemwide

The first type of concern is expressed as **systemwide failure**.

- As far as mitigating the effects of system failures for this frontline work where you're basically using an entire computer system at a site, one idea is to move away from the batch-queuing model. Move to a model that is closer to a physical experiment, as if you're using, for instance, an astronomical telescope. In other words, it would be much more beneficial to us to be able to run for a long time, but to book that runtime at some point in the future, and to have systems staff on call when the reserve timeslot begins, to fix anything that occurs.

- At 60,000 processors (or whatever it's going to be) I suspect that computation at that scale will not be possible using the current approach to batch production. How on earth would you assure users that when their timeslot came up that every single component of the system was functional? And how long would it stay in that state, given that the mean time between failures is proportional to the component count?

- Having jobs crash, often due to a node going down – that's the most frequent reason that a job fails to complete, having to fix things up.

- I'm running in this 2,000-4,000 CPU range at the moment. And within the next year we expect that to go up to at least the 32,000 CPU range, if not a factor of two more than that. The unreliability that I see in filesystems, even in batch-process launching systems, disks, monitoring tools – you name it. Nothing really works reliably at the 2,000– or 4,000-processor level today. I am extremely doubtful about it working at a level ten times greater than that.

- I've heard from both OSG and from the LHC Grid and from a number of other Grid projects. They all give roughly the same answer: somewhere around 25 percent of their remote job submissions fail. This is a shockingly high number. In general they don't know why the jobs fail – they can only guess why.
  The top reasons cited for failure are basic authentication problems. You know, the user might not be in the right gridmap file. There are also disk-related issues such as running out of disk space during the act of staging in some input file, or they don't have the right permissions. But then there are a whole lot of other failures that fall into the unknown category.

- It seems like always somehow NFS is involved in some very sticky way when we're dealing with GRAM2, and it's not a pleasant experience. The biggest hurdle that we overcame to get to where we are now is by throwing hardware at the problem and getting a BlueArc NAS server, which is a very, very high capacity NFS appliance. Before that, NFS was crashing more than monthly, triggered by the kind of NFS activity that GRAM2 does. We still crash every once in a while – maybe once every other month or so – but nowhere near as bad. We have some ideas what's triggering the failures. In short, GRAM2 is doing hard links across NFS, and either the NFS client-of-the-day or the NFS server-of-the-day is not always reliable enough to implement that right.

- Stability for me, in the context of filesystems, means running without failures. Servers don't hang. User jobs run to completion. So right now our filesystem hangs and jobs have to stop because they can't write data. We've got to get to the point that something figures that out, a backup comes into play, and the job can continue.

- The difficulty for us is not that things break, the difficulty is in detecting that something's broken. I may not even know who owns the site – it's just a black box to me - and something went wrong. Now I have to figure out what went wrong. So I do a little bit of probing and then either tell the remote site what went wrong or fix my stuff. Most of the time it is easy for me to figure out what is going wrong once it is detected.

- There're many ways a job can fail, obviously. One of the ways is that the job will be successfully submitted to a site, something will run, but not successfully. Let's say that a filesystem goes away while the code is running. To detect that type of failure is not that easy. This is because different error codes are returned, and some schedulers will say the job was successful while others will say that it was not successful.

- When everything is working, it's great. So a lot of what we need is more fault tolerance and improvements in the way errors and exceptions are handled. The errors can be due to hardware or middleware at any level.

- From a workflow point of view it's dealing with failures. If you can move 100 files with a batch script and they all got there safely but one, it takes you as much human time to deal w/the one that didn't as moving the 99 that did. So human intervention to deal with the failures is the expensive time.

- It would be good to really understand what kind of failures Grids like OSG experience today. Most of what I know is somewhat anecdotal. Getting a picture of this is a hard problem. I don't know that they know. TeraGrid is the same way. It would be nice if there were somebody tracking and documenting failures in an organized way.

- It's much more common that you suffer a failure in the first few seconds, than it is the last few seconds. If any node, for example, can't see the parallel filesystem, that's fatal to a user job, but it might be something that can be fixed quite quickly by a sysadmin. But if you're running in batch, you wait days (if not weeks) till your batch job starts, it fails instantly, and you have to go through the whole thing again. So the operation of these things needs to be made reliable in both physical and human terms. You've got to have systems support to overcome that kind of problem in real time.

- We have so many things to keep track of we're losing the ability to keep track of it. We're building tools that need the information to function; they need the information to leverage the infrastructure. Tools need the information to help the users get the work done. But the systems that provide the information are breaking underneath the load. Then all these tools and infrastructure become unworkable and work stops.

3.3.1.2 Service level

The second type of concern is expressed as reliability at the **service level**.

- It should be a stable service to our user community. A reliable deliverer of services. This probably means we shouldn't use the development versions of the software tools. We probably should use only the production versions. If the production versions are compatible, it will be much easier on the application developers.

- GRAM2 has been more stable and reliable than GRAM4. That is the only reason I prefer GRAM2 over GRAM4. I need at least 70% success rate to consider a service stable. Ideally we want it to be much higher, but with GRAM4 we are seeing a much lower success rate. I certainly don't want to blame everything on GRAM4. We've seen hardware failures on the cluster side. But I would say GRAM should improve the way it responds to hardware and network failures.

- I personally would like to have many of the basic services, like GridFTP and GRAM, be more reliable before I see more features coming out.

- Not all the tools work as well as you would hope in terms of doing what they say they'll do, or having bugs, or "oh we didn't think of this yet." So there are a lot of maturity issues with some of the tools we try to use.

- The ability to pull things together so easily now using a Web interface is cool, but at the same time this opens up problems because it really allows anybody without a formal background in software development to develop these applications. The concern is similar to my pet peeve with Visual Basic: the resulting code seems very fragile and you have to be very careful how you use it. Things break, or the maintenance of those types of applications become a nightmare at times.

- The goal I'm really looking for is to have a mean time between failures of three months.

- Give us the hooks to make it redundant if you don't do it yourself.

- Let's say you decide that within the software you're developing you want to rely on this particular open source software that seems really cool. You need some assurance that this piece of software will have longevity before jumping into it. Or you may decide that you are better off writing it from scratch, which you really want to avoid if at all possible.

3.3.2 Diagnostics

In user interviews conducted thus far, observations related to diagnostics fall into two categories.

3.3.2.1 Error messages

The first category is interpreting **error messages**.

- When these problems are happening, for instance when hardware fails, the middleware we rely on gives cryptic error messages that we cannot read and parse automatically in order to adapt to it. The GridFTP "login incorrect" error does not provide us with sufficient information. In other contexts the source of login problems typically are on the client side, but in the GridFTP case it is often a server side problem. So what I would wish is when middleware cannot determine the particular error (and it's reasonable that it cannot determine everything), I would rather it propagate the original error message. Send it up the middleware layers of the architecture, instead of misinterpreting something and issuing a misleading error message.

- Another type of information that is lacking is documentation about errors. For example with GRAM, all we get is an error code and there is not enough documentation explaining the error. We then have to google to find out how other users handled the problem. Sometimes we even need to go as far as to dig into the GRAM source code to determine under what conditions the error code is sent. There is some documentation, but not at a level enough that we can use it.

- Sometimes we get weird errors that don't really reflect what's going on. Other than that it's fine. Weird errors like "Error code 17" that supposedly means one thing, but is most commonly due to something else. For example, it says, "could not create job description file," when the real reason is the user doesn't exist.

- We can't use a tool that produces a three-kilobyte file of error messages when something bad happens. Something happens and then all the processes start sending messages saying, "I can't do this. I can't do this." We can't manage that. That's too much information, which is just as useless to us as no information.

- If we see certain errors right up front, we cannot directly take that and send it to, for instance, the TeraGrid helpdesk. I have to do at least an hour of digging, because if I send directly an error message to the helpdesk, they will reply, “This is something to do with your client side. There is something wrong.” So I dig deeper and deeper and go through my usual tests, and see, “Oh, this service is down. Ok, here is what’s happening.”

- Their error messages are very cryptic. The most common error we get is “login incorrect,” but it has nothing to do with an incorrect login. It’s something like a hardware problem, or there’s a node goes down in a striped GridFTP server, or the allocation is out of the limit, or some scheduler is paused. For all these conditions we get the same “login incorrect” message.

3.3.2.2 Troubleshooting

The second category of observations related to diagnostics involves **troubleshooting**. Users indicate a lack confidence in dealing with runtime problems. The statement that troubleshooting currently requires a deep understanding of implementation details is particularly noteworthy.

- How you go about troubleshooting when things don’t work? Example: So I do a globus-url-copy from one center to another and I get an error message saying “End of file encountered.” And the file at the other end is of zero length. Now what do I do? Right now, I send an email to the administrator asking, “What does this mean? Why didn't it work? It worked six months ago.”

- Solving problems is easy once you have all the data in front of you. It’s getting the data and knowing what data to get that's the hard part. Networks are notorious for this, right? They’re black boxes. Very rarely are you lucky enough to have access to somebody who can actually find out operational status on routers and the like. So you have to infer what’s happening by using things like Iperf, netperf, or pipechar.

- We’re worried about things like firewalls and security policy getting in our way. I don’t know if security policy is a technology obstacle or not, but it could be considered one. Part of the problem with logs is that there is potentially sensitive information in there, and if you strip out the potentially sensitive information, you often lose the ability to do troubleshooting.

- When anything goes wrong with your certificates – site certificates, anything like that – it’s completely beyond the scope of anything a user can do. And usually it’s beyond the scope of what the computer center personnel can deal with as well. It usually means that you’re just crippled for a couple of days until the one guru at site X can actually figure out why what used to work no longer does.

- When you write multithreaded code, bad things can happen: deadlock. Another condition that also appears in serial code, but is perhaps more pronounced in multithreaded code: accidental memory overwrites. A tool to handle that at large scale would be tremendously useful. By large scale I mean hundreds of distributed processes – even thousands.

- When new software tools become available, it would be useful to know what the new features are, whether the features are compatible with previous versions, and where the incompatibilities might impact the other parts of the system – because in this case we'll know what to troubleshoot.

- Finding out what to do next or troubleshooting is not something I am capable of doing – not at this point, without going through a learning process, which I don't have time to do.

- The bigger missing documentation is in the troubleshooting area, when something is happening and we need to find out how to deal with it. Troubleshooting-type of documentation is not only for me but for system administrators – they struggle without this., because whenever something happens, we immediately post it to help@teragrid.org, and the system administrators try to figure things out.
  All the troubleshooting right now requires knowledge about the internals of the software. So only experienced people can troubleshoot right now. So if expertise is missing on the admin side, the issue keeps spinning for three or four days.

3.3.3 Communication Issues

Four types of communication issues have been observed thus far.

3.3.3.1 Globus developers

The first type of issue concerns **how Globus developers communicate** with the community.

- The other thing that is a little bit difficult: sometimes I think there's a reliance on the email lists for archiving information. And that's great, because sometimes the details really only exist in an email list and you want to be able to find them. But it can be hard. There are so many email lists I'm trying to monitor.

- It would be helpful if some of the campaign details again were in a more centralized place. Information that was exchanged through the email lists that's pertinent to the roadmap or the campaigns could end up in this other place.

- It would help us to know the assumptions that Globus developers are making on the various files. I'm referring to what are they doing with locking and where the state of the gatekeeper is living (for both Web services and pre-Web services). I know the broad strokes, but we'll need to know a lot more detail when we do the redundancy work. We'll be emulating the service, and we need to know as much as much of nitty-gritty implementation details as we can. Right now we find these things out by trial and error. Take our work with our job forwarding as an example: We took a file that lives down in a Globus library, condor.pm, and rewrote it to do what we wanted it to do. We're basically emulating the pre-Web services 2119/jobmanager-condor interface [*fragment of the conventional network address for remote Condor jobs*]. If you send a job to that on our Grid, it's not really Condor underneath the covers; it is our own proprietary system.

- Sometimes when trying to work with various Globus developers, I get the feeling that there's nobody really in charge. Everybody seems to say, "Well, I don't know. Is that more important than this? Is that more important than that?" And the developers are very hesitant to commit to anything without talking to somebody else first. One week it'll sound like it will be a priority, and the next week the work will get bumped. From the outside perspective there doesn't seem to be a lot of cohesive direction and vision. It seems like a lot of firefighting and jumping around. All the Globus developers we've worked with have been great to work with. But every single time you ask, "Hey, can you add this?" they'll say, "Well, sure, but I've gotta find out if this is more important than that." I always get that response. I'm talking about tasks that take somewhere between a half day and two days.

- The Globus team has gotten better at this, but there are still times where the team appears to be self-focused or focused inward. This doesn''t apply across the whole team. But some folks seem to be focused on infrastructure for infrastructure's sake, as opposed to infrastructure for other people to build on. There are still some pockets of that occasionally. But that's certainly not the rule. As I think about it, the teams I've interacted closely with – the RLS and GridFTP teams – I can say that's the opposite. They tend to be very supportive in terms of reaching out, asking for use case scenarios and requirements, and being responsive to input.

- We're trying to be more deliberate about designing and thinking ahead, without going overboard, in terms of how the pieces fit together. I'd say that's going to be a larger component of what we do now: going out and talking to the different middleware providers to understand what their roadmaps are, so we can try to get an insight into what things are going to look like one year, two years, even three years from now.

3.3.3.2 General feedback opportunities

The second type of issue reported by the user is a sense that there **aren't general opportunities for scientific users to express feedback** about the systems they use:

- The way the centers work all the information comes down and there's no feedback – this conversation not withstanding – from the poor users at the end of this who are forced to use poorly designed and inadequately supported computers. And they suffer terribly in loss of scientific productivity dealing with the endless failures at every level of these systems.

- There is no feedback from the users to the center management or to the NSF, in terms of the cost in human resources in using these systems. The current round of the NSF program is a perfect example: this obsession with buying a petaflop computer for political reasons, presumably to brag about it internationally or something, with no input whatsoever from the userbase; no clear understanding of how it possibly could be used; and no input from the end users as to its architecture, its characteristics, or what it will support.

3.3.3.3 Perceived differences in worldview

The third type of communication-related issue involves **perceived differences between the users' worldview and that of computer scientists**.

- Another challenge is working with the domain science community and trying to understand their needs – trying not only to advance their science but also to advance your own. Because one of the problems we face as computer scientists is that we are seen as the technicians for the domain scientists. The advancement of computer science is seen as secondary, as opposed to something that could be an equal partnership. Establishing this type of relationship takes a bit of education on both sides actually.

- Because it's old doesn't mean it should be ignored. Ninety percent of the science codes in a recent Oak Ridge survey were found to be written in Fortran, for example. No one in the computer science community can be bothered to help a Fortran programmer anymore. They probably don't even know Fortran. But in science it's still tremendously important, and C is the next one behind that. We're not going to switch languages. I'm sorry to say that the DARPA HPCS language initiative is pie in the sky.

- Generally I find it hard to imagine that the people who do these security services have ever done any large-scale computing project. For instance, I'm moving files from Pittsburgh to NCSA, so every time a job finishes (and these jobs run for over a year, one after another), I have to transfer a file. The notion of typing in a password and doing that by hand is very annoying, compared to having it happen with some sort of automatic system.

- In some areas of science people use packages a lot, and they're used to the idea of typing in some sort of GUI and hitting "go" and getting some sort of answer. We're about as far away from that as you can possibly get. It doesn't even make sense to me to consider such a thing. When the computation time is measured in months, any kind of traditional view of that just doesn't work. For example, this one simulation that I'm running I started working on … in the last week of November and it's now June. It's not over yet. These things don't fit well with the computer science idea of running myriad little processes.

- Most of the services seem to do what I have been otherwise able to do for a decade or more (such as moving files with scp or ftp). So I'm trying to understand where the value is added. Maybe UberFTP is able to move my file about twice as fast. … I haven't yet tried it between Pittsburgh and NCSA.

- The people who are doing this ought to have some experience using these systems for large-scale projects. My feeling, perhaps out of ignorance, is that most of these tools that I've seen that are called Grid tools reproduce services we could do before. They seem to have complicated names and complicated protocols. And, regarding the security, the tools are not designed to run jobs that will run for months and months and months without too much user interaction.

- Some scientists can't distinguish between their domain science and your infrastructure: middleware, Grid logins, infrastructure, clusters, their particular science application, etc. Maybe they've even been given a science application from someone they're working with. They can't distinguish among them – it's all "The Computer" to them.

- When I go to a new computer at the Pittsburgh Supercomputing Center, I find they do an excellent job of organizing the information that I need to know in order to use that computer. In contrast, the Grid-related documentation for things like getting a certificate is organized into very small chunks. They weren't organized in the steps I wanted. There are lots of different acronyms. The documentation seems to have lots of different paths because there are so many different ways of doing things. As a user I want one way that is going to work, and work easily.

- For many middleware developers, they often think that's the only thing left to do after you install their middleware. We have to gently point out that application porting is a mere starting point that occupies a small fraction of our time, because it is in fact so easy. It is by no means the end of the story. If we just have someone who has a computationally intensive application that needs a lot of CPU hours, then we hook him up with our Grid, TeraGrid, or OSG, and we're done. But very often there's a great deal of social interaction to be done. There's a great deal of organization building.

3.3.3.4 Intraproject

The fourth type of issue is related to **intraproject communication**.

- I'm a sysadmin, so this may not be true of everyone, but I find meetings, collaborating with outsiders, getting everybody up to speed, exchanging docs to be very, very time consuming. So ideally I can have everything in my lab and have my students stop by my office to answer their questions, write something on the whiteboard, and have them start running it immediately. That saves me a lot of time. I'm comparing this to bringing different groups together, having weekly or biweekly meetings, exchanging our little limited views of each other's work, and trying to make sense of how we are going to put things together. That seems like a big, big time drain.

- I, like my a lot of my colleagues, spend entirely too much time in meetings, on telecoms, and answering email. And not nearly enough time being able to just sit down and solve the problems I need to solve. When you're in these collaborations, there are just so many people you have to coordinate with, and it just takes so much time; it can literally eat up a third of my time. It's not all bad, but there are days I wish I could just lock myself in my office and do nothing but write code, because I'd get a lot done.

- In a project as distributed as OSG is, you spend a lot of time in meetings and writing emails, just communicating issues back and forth. So I think that's my main problem. If you want to be a part of something, you have to invest a lot of time. But the big problem is how distributed the project is. There are so many sites, there are so many projects and experiments, and they all have slightly different agendas. So something that might be important to you, nobody else might care about (or the other way around). Or you're getting pushback from people on something that doesn't make sense to you.

- When you start coordinating with groups like OSG, it's pretty complicated, because they're such a big organization with so many different conference calls. It's hard to figure out. Also just trying to keep up with them in e-mail lists – some of these lists get hundreds of messages a week.

3.3.4 Technology Adoption

In user interviews conducted thus far, issues related to technology adoption fall into seven categories.

3.3.4.1 Difficulties using national infrastructure

The first category pertains to difficulties **using deployed national scientific computing infrastructure**:

- I find it very difficult to figure out how to register these certificates at different sites, because I have a different user name at NCSA and at Pittsburgh and at SDSC. So first of all I found it difficult to find the right place to start looking for documentation about how to get my certificate registered at a new site. I found it easy to google and figure out how to get a certificate. But then to get the Distinguished Name registered and hooked up to each individual account took me a long time to find the right place to start looking for documentation to do that. And then once I found the documentation, some instructions said to use gx-map, while other places said to use gx-request. In almost every case, neither one was on my path. I had to hunt for probably thirty minutes before finding it so I could actually use it.

- I find that I often don't have the right commands by default in my path.

- I know that some of the large bioinformatics applications, like in CAMERA [*Community Cyberinfrastructure for Advanced Marine Microbial Ecology Research and Analysis*] projects, they have amounts of data that are 100 times larger than what we deal with, and they have no idea how they will deal with that. We were trying to download at least small chunks of their data and it was taking hours and hours. Sometimes we can use GridFTP, and sometimes you can't, because in some cases we don't have a GridFTP client available to us.

- Once I get a transfer to Oak Ridge running right, it's pretty much fixed, except for congestion on the network (like somebody kicking off a big job at the same time I did). That is, until the next time they do a kernel upgrade and blow away all the modifications you made.

- One of the big things keeping GRAM2 alive is interoperability with European experiments. They are not going to GRAM4, as far as I know.

- Security is out of your control and requires bizarre and completely Byzantine communication between centers. For example, try using an NSF certificate at a DOE site. Dead on arrival.

- There are no obviously robust methods that we use to help us get around this. Now, I believe there's a thing called Reliable File Transfer (RFT) that might help. But again, we don't just use NSF TeraGrid, we use DOE centers as well, and it's not clear to me that they would implement anything like that.

- Typically it's difficult to go to a regular Globus site and just run our code because we rely on so many external libraries and tools that are not part of the Globus standard install. Sometimes it's possible to request these things to be installed at the site, sometimes not. If it's not, a lot of these tools can be compiled and installed in a home directory and run from there, as opposed to assuming the system has them.

- UberFTP: the major challenge I faced is the first time I tried to transfer a file, it only transferred one tenth of it and then it stopped. And in general it's not always clear who to ask for help because it's always a transfer between two different sites. So you have to get both sides involved, which can sometimes be difficult. Sometimes they don't communicate with each other – they'll only communicate with me. They may have different help systems.

- What I find annoying is there are so many authentication schemes. For instance in our DOE SciDAC-funded project we have computers at Jefferson Laboratory, Brookhaven National Laboratory and Fermilab. Everyone uses a different security system.

- When it comes to GridFTP and moving things in and out, we only control one end of the transfer. We can make sure the machines on our end are beefy enough and are configured correctly and tuned right. But if the guy is trying to transfer the other end off his laptop, we'll only go as fast as his laptop. Data transfer is an interesting problem in that respect. It's a two-ended problem. If you're trying to schedule the transfer, it requires co-scheduling – you must schedule resources at both ends. You don't have control over your own destiny: you can control your end and you can coach the other end. But if they don't have the hardware, there's nothing you can do. And that actually gets quite frustrating. While I know that rationally they understand it, all the user knows is he's not getting what he wants.

- If you email them about GRAM2, they will be very responsive, but if you email them about GRAM4, their response is only best effort. I think it is this way right now because GRAM2 is the production version for OSG. If GRAM2 is failing for them, the site is considered to be failing. If GRAM4 is failing, it is not that big of a deal for most people.

- Metadata very quickly becomes very application specific. And most scientists have perhaps not as good a system as they would like, but they have a system of some kind that they already use for tracking metadata. So we don't provide a standard system service that could be called a metadata service.

- Keep GRAM2 around in addition to GRAM4, especially in the Open Science Grid at large. The Open Science Grid doesn't have an information system to reasonably send GRAM4 stuff around. Our whole information system right now is tied to GRAM2.

- The file transfer I was doing from NCSA to Fermilab is going to a special tape archive at Fermilab that's managed by dCache. And so, to make this work, I have to use an SRM-copy. And the SRM-copy was failing. And the reason it was failing is that Fermilab has to set up certain map files to make that work, and those are not being properly maintained. Maybe it's that not enough people are doing this kind of thing, so these things are not being maintained to the point that it makes it easy to rely on whether or not it's going to work. So then you finally just fall back to the old FTP again, and that works, sort of. But in order to get the files onto the tape archive at Fermilab, the scp has to go through two stages. You have to move files from a disk to a disk; then you have to move them from the disk to the tape. So that's a painful process and doubles the amount of work.

- There are the hardware problems: dealing with the sluggishness on some of the networks like the ESNET, which we've had some problems with recently. The file transmission rates are painfully slow, errors occur, and then we have to retransmit.

- Users of our project must negotiate separately with each site they want to use. So that means they need to contact the person at each site who has the power to authorize them. This is hard at a lot of the sites because they're set up to serve their local campus users. It's easier for TACC because being a TeraGrid site, TACC can say, "Sure we'll give you a little starter account, and go through TeraGrid to get more cycles."

3.3.4.2 Cost of use

The second set of adoption issues concerns the **perceived cost of using the technology**:

- I would not consider installing Globus myself. I don't like the overhead. When anything goes wrong with your certificates, site certificates – anything like that: it's completely beyond the scope of anything a user can do.

- GridFTP carries all the baggage of Globus with it, but it's the only component we're interested in. Really it's just an FTP program – why on earth do we have to bother with all the certificates and all the stuff that goes with it? All we want is point-to-point transfer to be fast and reliable.

- If I could be convinced that a non-GSI version of GridFTP was stable and secure, I'd use it. I hate GSI. It's very good at what it does, but it is a pain. When I was involved in GridFTP development, GridFTP didn't have problems – GSI had problems. Once I could get people past the GSI issues and get it all configured, GridFTP just runs.

- One time I had an expired Grid certificate, and at NCSA it was quite easy to generate a new Grid certificate, but only because I had taken a (paper) file folder from my office, which normally I would not have with me, that has my default password (because I'm traveling right now.) So if I hadn't decided at the last minute to take this file, I would not have been able to get a new certificate.

- Running a CA, deciding whom you trust – that's all a large pain, a very large pain. For example, you have to get a CA certified by TAGPMA and buy special hardware. And to be blunt, after all this is done, as a user we don't gain much of anything – no additional capabilities: you can access the same machines as you could before. You know, it's a big hassle for some potential benefits, like delegation, having your own agents out there to do things for you. There is some potential there, but it just hasn't come to pass that we've needed it.

- If we were to start using all your RPCs, your Secure MPI, and the secure-wrapped standard libraries that have been modified for Globus use. If we were to use all that plus GridFTP, sharing a common authentication infrastructure, then the burden of all these additional layers and centralized key distribution would seem worthwhile. But if we have eight nodes, and we just want to launch our scripts remotely, that seems like a lot of work.

- The time burden associated with setting up the application-specific environment on the remote machine is a big challenge. The University of Chicago Grid experts handled the Globus-specific setup, so I can't comment on challenges associated with that.

3.3.4.3 Ease of use

Ease-of-use issues have also been described:

- It's not like they can go to the Website, download a Microsoft installer package, do a double click and the software installs and you can start it. It's not working that way, and it will probably never be that way because you have to have credentials being created.

- The complexity and reliability of the tools we have to work with are a key problem for us. If you add up all the tools, you don't get a very good user environment out of them. It just still seems to be too hard for our users. For example, there's only been one person I've worked with so far that can really just figure out the stuff on his own. But he's really an exceptional kind of person this way.

- The management of credentials by users directly is too difficult. Our user base is not sophisticated enough to manage their credentials directly, so we are moving away from that approach. We'll be beginning to rely on MyProxy and similar types of credential repositories so users don't have to manage their PKI credentials themselves.

- The VDT is very helpful as far as getting things deployed much easier than in the past. But then after that, trying to get users working with those deployed tools is still a problem, and takes a lot of our time to help users. So ease of use is still a big problem.

3.3.4.4 Cultural barriers

The fourth category contains observations about **cultural barriers** that must be overcome:

- Sometimes there are terribly, terribly intrinsic issues to deal with. For example in the petroleum engineering field we find they have extremely powerful, well-developed expensive codes that the providers are happy to give you almost free academic licenses for. It's really shocking how open they are with their code – you can download it almost like you would a piece of shareware, extensively developed code. But then if you turn around to a particular researcher and say, "Ok, let's put this on the Grid," you find that they stop like a mule at a door because they won't let go of their data. It's the data that's important in that field. The data is highly proprietary, having to do with detailed field measurements of oil-bearing strata. They are absolutely unwilling to let that part from what their perception of what a secure space is. So we have to spend a lot of our time working with them to assure them about data security and implementing tools to make sure that they always feel in control.

- The problem is that you have to overcome the sociological barriers to sharing data within the medical domain; you also have to overcome the issues with the law because patient privacy must be protected at all times. And if you can solve these two issues, you can aggregate information that is very, very relevant – and can outpace efforts at the National Institutes of Health because many of these clinical trials accumulate only a very small numbers of cases.

- The same people who are probably logging on with cleartext passwords to POP email accounts react with great skepticism when you approach them with an absolutely locked-down X.509-secured, strong cryptography solution for controlling access to their data.

- There’s tremendous chaos in the identity management area. Everybody thinks they’re in charge of identity management – everybody! It’s like when I first started teaching, I went home and told my wife, “Everyone thinks they’re my boss: students, the dean, my funding agency.” The problem is that none of them are wrong. Certainly your university thinks they’re in control of all of the computer identities associated with you. The virtual organizations that you work with all want control. EDUCAUSE and Internet2 think they’ve got a good scheme. TeraGrid has its own thing; … they want to be able to decide who in your university can log on to their resource, and they’re not interested in your opinion about it.

3.3.4.5 Technology integration

The fifth category includes adoption issues relating to the **integration of new technology**:

- “Keep it simple” would be the only real advice I would have. You know – the KISS principle. Users these days have got an unbelievable amount of extra work to do compared to the supercomputing programs twenty years ago, when all you needed was a Fortran compiler and a Cray XMP and you were absolutely the best in the world. The complexity of it now is so great that I see it breaking down. It isn’t worth our time to consider adding more sophistication. If we’ve got any spare time or any spare brain cells, we want to use it adding sophistication within our code in terms of things that are domain-specific to us.

- An important consideration is which tools are extensible and allow me to build on top of them. This is in contrast to tools that try to provide a complete solution that force me to rip and replace stuff. I try to stay away from the rip-and-replace tools because we just can’t do that. Such tools offer solutions but require me to give up other stuff that I’m already doing in order to use them.

- Educating my home institution about the Grid infrastructure itself is important. I spend a fair amount of my time making sure that things we need to implement to make Grids work aren’t going to get tripped over by the folks who do security. This is, of course, a very common theme. We spend a lot of our time making sure there’s adequate communication there so that nobody shuts down my Grid servers because they don’t have username/passwords expiring every ninety days.

- I can look at a class of tools that does certain things – purporting to do something, for example “manage data transfers” or “manage workflows.” I then try to decide if the tool offers bright shiny new functionality but will, at the same time, be unstable and I won’t be able to rely on it.

- I have mixed feelings about Globus as it is. It forces the user to implement their code in some very specific ways. So there's a certain mindset that you have to work with. You cannot just take your code and just pop it in there if you really want to take advantage of Globus. Otherwise, you're just putting your own code on Globus using your own socket code, disregarding Globus security, and you're just using Globus to schedule things and gain access to machines. So unless you do it the Globus way, you're not really utilizing it.

- I look at the interfaces. I'm going to usually have to put some glue in place to pull these pieces together in reasonable ways, and how easy is it going to be for me to do that gluing? Do I have an API that's only supported in one language? If it's not my first choice for the project, I'll need to extend outside of our area of expertise. Or is it something that's technology or API agnostic, and I can easily just write whatever I want?

- If there will be some continuity between the releases, that would be helpful. Ideally we would not need to rebuild everything in our system to accommodate the new changes. It would be really good to somehow lighten the burden of transitions to new releases.

- It's wonderful when new things are developed; but every time there is a new tool available, it means that in a while we will need to rewrite the whole system to accommodate the new release. And this can be a problem. I understand that new technologies are being developed and that's why they are getting better and better. But it's a little hard on the application developers when new versions are not compatible with the other parts of the system.

- Most of our applications actually don't code against any APIs. They need to have the environment and security managed for them. So I can't go to a data analyst and say, "I need you to link with this library so that your tools will interact properly with the security." They expect the infrastructure to operate at a level either above or below that, depending on how you characterize it. They just want to run their job, and they want everything to be handled for them.

- There are a lot of different bioinformatics tools, and currently we are mostly installing them locally to run them. Sometimes we are submitting it on the network, but probably we just need somehow a more developed system in bioinformatics for Web services. So it should be probably Web-service based, the whole infrastructure. Note that we don't have any distributed algorithms. All of the data and all of the parallelization we are doing are embarrassingly parallel.

- To me, Globus is a set of daemons and infrastructure that
    - provides a unified security mechanism with cryptography, key exchange, and authentication on each service using a common set of keys;
    - provides a uniform remote procedure call interface;
    - provides some file transfer protocols using multiple underlying network protocols;
    - has some scheduling capabilities (I guess limited to per node scheduling); and
    - contains a set of standard libraries of tools that one can rely on being available on Globus nodes.

  In order to start doing something outside of the Globus-provided services but staying within the Globus security network, one has to learn additional APIs and how to code things up. So it seems like there's a high startup cost to use Globus. To me it's cheaper to put 20 CPUs behind a firewall and a private network with no way in except through some gateway node that's well secured. I can then just run whatever I want behind that firewall using the most approachable, easiest to use toolkits with the least overhead and with least restrictions on how we code things up. I know there are many people who truly want to distribute their processing across multiple data centers. To them security will be more important. But as long as our project will fit within our local cluster that we can handle ourselves in the back of our lab, it's too much additional work to do it the Globus way.

- We use a lot of libraries and toolkits like R, which typically are not included in a standard Globus install. So we can't rely on them being on other Globus clusters. So we're shipping our own precompiled code that maybe has R installed in the home directory. We do things like that to get our code running, but in the process we're missing the point of Globus.

- What kind of logging does the new tool have? Is this a tool I'll be able to drill down easily when I think there's something going wrong? Can I turn up the logging levels so I can really get a picture of what's going on?

- From the security realm: there are a number of solutions based on proprietary tools. Some people are interested in those because they seem to offer to the users a better experience. I use the term "seem" because I'm not convinced that they actually offer a better experience for the user. But the problem with the integrated solutions is then on the backend there's no choice about how to hook them in to other systems and services.

- The attempts from Globus-related teams (I don't think these are Globus Toolkit proper) to provide tools and infrastructure to help with metadata and provenance have not scaled. And especially in terms of provenance, they've required too many application-level changes. The approach was "Just do everything this way then you'll get the provenance information." But there's no way to "just do it this way." That's not the way my users can be approached. They are going to do their science. The science is going to lead, and all the other stuff has to be tacked on.

- The things I need to do from a syntax perspective are completely different between GRAM2 and GRAM4 and require a rewrite of my stuff. And the RSL versus the XML is completely different. All that stuff is completely different – but functionally, no.

3.3.4.6 Lack of knowledge

The sixth adoption issue is **knowledge-related**:

- I think what really needs to happen is to give training courses to hospital IT on what is Grid technology in general and what is a concrete implementation of it. That would really help because then we wouldn't have to repeat these discussions with every single institution when we do a deployment. But my gut feeling is that it may be a little too early because this whole field still has to mature – not only in the Grid domain, but especially the interaction with the clinical hospital domain and the overall healthcare enterprise.

- I would also find more tutorial-like information quite useful. For example, I read the whole Globus Toolkit 4: Programming Java Services book, and I practiced a lot of examples in there. This is kind of helpful, and we would like more examples – like when we are writing clients to a GRAM service, we look for more tutorials or even CoG help in some sense.

- It is difficult to figure out where to find the right documentation. Once I do find the documentation, it's very hard to understand. It's full of acronyms and refers to unfamiliar and unnatural concepts.

- It seems to me that in order to get Grid solutions you have to be pretty tech savvy. Getting the certificates, doing the job submission, doing the DAG of the workflows on Condor, managing the security: all of that seems to be an enormous barrier for actually getting jobs done on the Grid. It didn't seem to me like there is any mechanism on the TeraGrid or OSG to sit down and work out how to solve problems together.

- People write great developer guides. And that's great for somebody who is a developer. But what about the rest of us? What … does this thing do? Even with GridFTP we've tried, but people sometimes just don't get the big picture. Concepts as simple as "What's a client? What's a server? What's a third party transfer?"

- The learning curve for some of the software is quite hard. It takes me five months to train people to use some of our domain-specific software. That's a long time. But I'm not sure it's related to the software/technology. I think it is that some of the concepts are difficult.

- They say, “Oh, good. You’re here. You can do this now.” And you have to say, “No! That’s not what we’re here for. We’re happy to help, but you have to keep doing your science.” The group is populated by a mix: people who are just getting started, people who haven’t figured it out yet, people who really don’t know how to use the tools, or the senior professor who is now going to ask you from this day on how to log on to his email. So more than half of the time when we try to work with people who ostensibly are researchers and scientists in their field. They say, “You’re so much more qualified than I am to do this that it’s hopeless for me to do more.”

- We’re working to convince the radiologists that this is a good paradigm that is beneficial to them. And the radiologists are not technically at a level where they understand what the underpinning is. But that’s not really necessary. So the approach we’ve taken is kind of the soft approach of learning by doing. For the sites, if they see “oh, this is working,” or the security model is gaining trust, then you build confidence. After building confidence you can go to the next step.

- We’ve certainly hit challenges with the road to Web services: writing Java submission scripts that can be submitted through Web services. The XML – that’s the place we sit down with the user. We’ll just do it with them because many people look at XML and, well, it doesn’t fit their worldview. But it’s utterly trivial to sit and work with them, saying, “This is how you specify the batch queue.”

- With Globus there’s lots of information that you need to get Grid certificates:
    - Knowing which one is the best one to get
    - Knowing how you use those certificates to authenticate
    - If you’ve gotten one from somewhere, how you get to another place and get authenticated there

- If someone were trying to do this who didn’t know the area that well, it would be kind of tough. For example, compare it to something like Linux. You can take one of a number of distros and then kind of do all your shopping there. There’s a bit of that going on in the Grid area, but I think not everything needed is covered in them yet. So, for example, metascheduling: we couldn’t go to a distro like VDT and pluck it out of there. It’s not in there yet. It’s not mature enough. There hasn’t been a consensus yet on what is the best one. So this is where it’s a good thing that a number of us on the project know the area. We know the providers, contact them, and work with them directly.

- I’ve been trying to hire a post-doctoral fellow who's a neuroscientist to do computer modeling for two years. So one technologic obstacle is training more computationally sophisticated biologists.

- The technology challenge that I face is the learning curve involved in using different software.

- There are still not enough people in the world, as far as I'm concerned, that have real in-depth Globus knowledge. And certainly they're hard to find and hire. So we train people up here, to the best that we can. But it's still hard to say to someone: "I'm thinking about putting RFT into this service, but I need to understand where it's going to break. I need you to stand up an RFT, and throw larger and larger requests at it until it breaks." Or, "I want you to throw larger and larger requests at it and tell me the load and memory footprint on the machine." I could do that with my staff, but I usually end up getting it kick-started and spending a little more management time than is optimal. That is not a comment on my staff, because they're all good, hardworking, smart people. They just don't have some of the expertise, especially with of the Web services stuff now in Globus Toolkit 4.

- There was a case in which I was creating configurations [*simulation results*] at Pittsburgh and doing analysis at IU. This is a perfect example for distributed computing: where you run a job at one place and you put the output somewhere else and run some subsequent step of the job. I could do all that just fine with *ssh,* using the network, queuing a job at IU. Nobody ever picked up the challenge of trying to do that with Grid commands. All I needed was a little background script running at Pittsburgh that I could understand quite well.

3.3.4.7 Support burden

The seventh adoption-related category involves the **support burden associated with the technology:**

- I solve the engineering problems, and I do everything else that's needed to support this lab. I'm very short on time. If I have a choice between using local CPUs to do work or Globus CPUs controlled by somebody else to do the same work, I'll use my local CPUs. I know it'll take me an order of magnitude less time to set up something on my nodes to give the numbers that need to be computed than it would take for me to schedule the use of the Globus nodes on the TeraGrid (for instance). The setup work I'm referring to includes account setup for the students that need to access the resources, scheduling time to run our code on them, and installing prerequisite software. To use the remote resources, I need to either work with the admin of the remote sites to install things or devise instructions for the students on how to compile the software in their home directories so they can be ran that way. Or I can just use my nodes and just get it over with much, much quicker.

- If we can get a lot of users going on our clusters with minimal interaction with user support staff, we should be able to do as well when getting them on the Grid.

- Monitoring our infrastructure decreases my productivity. We have a big academic course in the spring, for example, where my research goals and development tasks have to be ignored because the burden of supporting the tools is great. What I mean by supporting the tools is trying to see what's happening on which resource.

3.3.5 Specific Technology Issues

We collect here a number of comments about specific technology issues.

Issues relating to **C WS Core**:

- C WS Core: The examples and documentation – I know they're working on that and it will get better. But right now there's not as much documentation and not enough examples.

Issues relating to **GRAM**:

- A lot of times cluster users will modify their *.profile* to set their environment for their jobs. They want those values to be used for the job via GRAM4, but they aren't. In contrast, if they submit the job directly to the scheduler, those values will be picked up.

- Both GRAM2 and GRAM4 are lacking in the same thing, and that's the ability to do co-scheduling. That's the biggest problem for us. Both GRAM2 and GRAM4 are great for saying I need 10 nodes on that machine, and I want you to run this application when you get them. And I don't want to worry about specific scheduler syntax. I don't care. I'm just going to specify the job in XML and let GRAM talk to the scheduler for me. GRAM is great at that, negotiating to put you in the queue, notifying you when the job is running, etc. That's perfect.
  But we don't run jobs like that. None of our MPICH-G2 jobs run like that, meaning on a single machine. All of our MPICH-G2 jobs necessarily run on two or more machines. It is imperative that the jobs are co-scheduled: that each job is launched near the same time. It does us no good – in fact, in some cases it does us harm – if one job actually gets through the queue and executes on machine A, and then two hours later the second job gets through the queue and begins running on machine B. It doesn't do us any good. We need to make sure that they both get through the queue and hit the nodes at the same time.

- Documentation could always be easier to find. In the effort of deprecating GRAM2, the Globus documentation has been made very hard to find. At least it was the last time I looked a few months ago – I haven't even checked recently.

- GRAM4 is a huge resource hog. It takes 700 megabytes of memory to sit there and do nothing.

- The biggest concern for GRAM4, however, is the GRAM container goes into hibernation or stops for a while without any log messages, and it just comes back by itself after a few hours.

- The challenge that we have to solve eventually is to figure out what the GRAM4 analogue of the GRAM2 forwarder will look like. How are we going to implement in GRAM4 what we've done for GRAM2 for our Grid: Will we just put GRAM4 in front and keep GRAM2 in the back? Will we try to do a GRAM4-to-GRAM4 thing?

- The other thing is Globus' nasty habit of (at least one time in three, and sometimes more) deleting the file you would like to see before you can get at it. This is with regard to debugging GRAM2.

- There are also issues that we have with GRAM, be it 2 or 4, with regard to job auditing. It always takes investigation into at least three log files to get a full picture of what has happened with a job. Not all of the authentication information is in the right place always. ... There could be more information.

- We just went through an issue that turned out not to be a GRAM4 issue but a Condor-G issue. It took us two or three weeks to debug that and identify the problem. It turned out that some authentications and authorizations didn't play nice together. Also, Condor-G was making calls when it ought not to (or not making calls when it should). So one GRAM4-related challenge would be working with the external callouts that are common in OSG.

- Many of the RSL attributes that are defined, if you can find them on the Webpage, are not implemented in the backend scripts. So, for example, we just added some memory support into ours here. Given that our nodes are multicore, we need to allow our users to say, "I need this many processes with this much memory per core." So that results in us putting one process per core per node, or perhaps one process per two cores per node, depending on the amount of memory they need. So that wasn't a big deal, but it was something we had to add in recently. The LSF.pm scripts did not include support for taking the min memory XML-based RSL attribute and turning it into the right LSF line in the submit script.

- Given our stakeholders, it's unlikely that we'll be rid of GRAM2 any time in the LHC era. I expect we'll have to keep it going for at least five years, maybe more.

- There is an issue when the GRAM4 state thing is mounted on a shared file system. This could really put a crimp in what we were trying to do with our high availability.

Issues relating to **GridFTP**:

- An engineering guide written for sysadmins (or people about to install a GridFTP server) is needed. There should be a document that walks you through the thought process:
    - How big does the machine need to be?
    - How big do the drives need to be? How fast?
    - What should the network connectivity look like?
    - Should I run a striped server? Should I not run a striped server?
    - Should I run GSI?

- One big problem we have is with the firewalls, with active/passive settings. We need to have different combinations of active/passive settings depending on the hosts. For instance, for some host-pairs we need to make the source active and the destination passive, but for others we need to make both active. So it's been really crazy, and we've had to do all sorts of hacks to switch settings.

- I really don't want to make a big strong pitch for GUI-based tools, but certainly in the area of data transfer that would make our life easier. So if I could get hold of the developers of CGFTP and say, "Make this real or make this go away," I'd do that.

- I tried to install the GridFTP client myself but it failed on Solaris, and then I gave up because I could use it from Fermilab. I didn't try to track it down further, but when I was trying to install it, it looked like it was trying to pull half of the Internet onto my workstation. Part of the problem was, I think, that I ran out of disk space.

- The interface to GridFTP is a bit clunky. We would like something to be as simple as *scp*. I gather that the TeraGrid project has done a fairly good job encapsulating some of the knowledge you need into tools such as *tgcp*, but I get the impression that some of those things aren't really maintained so well.

- The lack of a GUI-based client for GridFTP is a barrier to some of our users. We've tried this CGFTP thing that's coming out of China Grid in some highly incomplete state. That satisfied a couple of our users. Some of our users like GUIs, and they don't like using the commandline to move things around.

Issues relating to **RFT**:

- RFT is very good when you set all the optimal settings. On the default setting the performance is very bad compared to GridFTP. But if you tweak all the parameters we get the optimizations. So we need to find out and learn some external tools to provide these optimization values. For example we need to look more into MDS and see what are the optimal configurations to set between two hosts.

Issues relating to **information systems**:

- I essentially need to figure out:
  - Can I build my code here?
  - Will my problem fit on this machine?
  - What is the operating system?
  - What is the software that's already been installed?
  - Related but different: Is my prerequisite software installed?
  - The number of CPUs
  - The number of nodes
  - The amount of memory
  - The disk quotas
  - The scratch disk space

- It would be absolutely great if there were some information system – actually I guess it's probably not for Globus, because it's probably domain-specific knowledge. The information system would enable finding the services, finding the information, and somehow linking it in a simple way. In this case it could be distributed services and distributed data.

- MDS4 Index: It breaks, it's slow, and it's overly complex, in terms of the model. What I mean by that is:
  - XML and XPath is more than is needed 98% of the time
  - Java makes it quite heavyweight for small things
  - The last I heard, they were running in memory instead of out of a disk-based database, which hogs a lot of memory.

- The number of data products we are responsible for is growing quickly. Therefore the number of files is growing quickly. And for us the big issue is not so much the raw data size, because in a sense it's still a terabyte a day. But now, instead of being divided over a couple thousand files a day, now it's over tens of thousands of files per day. And they're all different sizes. We have to track so much information about the data now. And so, as has been the case for the past couple of years, the metadata is killing us.

- There are many monitoring tools out there such as INCA-based services, and the TeraGrid user portal has some of them, and the WebMDS is supposed to have monitoring information. But based on our practical experience, we see that the frequency of those tools monitoring GRAM and GridFTP is not accurate. So we end up having our own tools to test in real time whether or not GRAM and GridFTP are up and running. If they are down, we immediately blacklist that host until someone manually goes and brings them back up. Until they are back up, we submit to a different resource. The reason we need to test in real time is we have seen many examples where the monitoring tools aren't showing whether a service is really available. If you ping GRAM or GridFTP, it works fine, but if you actually try to do some functional thing (like transfer a file or submit a job), it fails.

- If I can find all my bank history in a split second, why can't I find a machine that meets my requirements in less than ten seconds?

Issues relating to **Java WS Core**:

- I think the Globus MySQL instructions and the way the database is connected should be revised. You have to install a specific version of the driver. Why is that? These kinds of things are a little bit nagging. And some of the drivers you can't even get anymore because they're outdated.

- The major problem with the Java container is that the database connectivity just sucks. In all the compilations (I started with 4.0.0 and then all the subsequent versions until the latest) the ODBC drivers always give me problems. Compiling this container can be a very simple thing, but it can also be a very painful, depending on whether or not you need these database drivers in there.

- More dynamic IP address handling is needed. You know, how the container handles the network coming and going needs work. I think the container's notifications could be a good fit for us, but we need something that works better in that environment. The use case I'm dealing with (in a different project) is where there's a sensor somewhere connected by a GPRS cell phone. The sensor gets different IP addresses every time it connects. It's just up for a few minutes and then goes down again. The current notification framework doesn't really work well in that dynamic scenario.

- There's also the issue of the guaranteed delivery of notifications. Let's say we have a sensor that needs to aggregate some data. So every now and then it pops up and says, "Ok, here's my data for the past hour" and sends the data to a service. At the same time it would check for any pending updates from the service, so it would process notifications sent by the service while the sensor was offline.

- The common use of PostgreSQL in the toolkit should be revised. I think MySQL is more common than Postgres.

Issues relating to **MyProxy**:

- I don't know where the documentation is, and if it doesn't work the first time, I have no idea what to do.

Issues relating to **RLS**:

- And as it turns out, relational databases are not the best way to model our data. We don't really use the relational aspects of it. What we really want are fast index hashes. So what I've asked the RLS developers to think about is abstracting RLS so that it can support other plug-in backends, just like the GridFTP supports other data storage interfaces (DSIs). I would like RLS to support different DSIs. It should have the relational database as the default, but also provide the option of using other methods of representing user data and its mappings between logical and physical filenames. Then what we would do would be to write our own backend based on a hash table approach, because I really like the RLS API and I like the model. I'm very happy with it as a service at that layer. What I want to get away from is the relational database backend because I don't think it's going to scale for us going forward five years from now.

- We tried to deploy RLS on a 64-bit machine, and during our critical production mode it did not scale beyond a certain limit, so very low scalability in 64-bit mode. We told the RLS developers, and they identified some problems in the C globusIO libraries. They gave us some fixes, and there is still an open bug report about it. In general the scalability issue has haunted us a lot, and so we've had to find workarounds on 64-bit machines. Things are fine on 32-bit machines.

Issues relating to **security**:

- It seems that if I were to have 70 Globus nodes under my control (which I have not reached yet, but if I had 70), I can foresee difficulties associated with centralized account management. Key management for Globus seems very complicated. Until I go beyond 20 or 30 nodes, it's been suggested to me just to keep the keys locally on all the machines and not try to centralize everything – it's much easier that way. Some heavy Globus users have suggested this to me.

- The standard Globus instructions basically lead the new user into an exercise of SimpleCA and building their own X.509 capabilities. These instructions are essentially useless in the context of any large-scale deployment where you actually have to trust each other and you need to build a foundation for trust.

Issues relating to **workflow**:

- Our Grid gateway uses VDL. We haven't transferred all of our domain-specific applications to VDL. Some time ago there was no recursion, but I think the issue may be addressed in Swift.

- VDL is good, but the problem with VDL was that it wasn't very stable. But then currently we are running pretty well with it. So I don't know what the future of it will be because I know that now it is called Swift.

Issues relating to **installation and deployment**:

- A lot of the more advanced configurations and uses of Globus seem to be not as well documented. So for me, that means I'm generating each key by hand for each user, and distributing the signatures to each node to allow the user to log in. It seems very painful and very complicated. And for what I was tasked to do (enable users to copy files and launch jobs remotely), it seems like a lot of work.

- A significant amount of user problems related to Globus or associated Grid stuff is simply that the client is configured incorrectly.

- Another technology-related obstacle I encounter is the issue of coherence of a given set of software. It is not possible to implement just one piece. Even all of the Globus Toolkit clearly is one piece. So the technology obstacles are ones of keeping the different components into a compatible state. In the two-and-a-half years of our Grid there has not been a time when we've had the latest software versions installed on all of our machines. We are still not up-to-date.

- So then when we go to add pieces like our metascheduler, if that falls out of synchronization with some features of the Globus Toolkit, it can cause us problems. Nobody owns these problems. We have to solve them because they're our set of choices of what to include.

- The challenge of maintaining a very big and very complicated software stack on more than 3,000 machines is very difficult. The solutions we have in place now for managing this are not adequate. I send the instructions to the sysadmins and they say, "What? This is crazy," and I say, "Yeah, I know, but it's all we've got right now." So getting a very complicated software stack distributed and running on all these machines is difficult. But this is mostly not a Globus problem.

- It is turning into a situation where you can't even use a distributed file system to get the software out there. There are more and more requirements, and more and more stuff has to be pushed out locally to every single compute node. Of course, the Grid was sold in a totally different way when it first came out. It was supposed to just live on your batch system host, and you wouldn't have to worry about it. The nodes wouldn't have to know about the Grid. In practice on the OSG this is not the case. The OSG stack for every single worker node these days includes all the Globus clients, such as globus-url-copy and the Web service equivalent. It includes Grid security certificates and certificate of authority files, which are used for authenticated file transfers. And then there are many more things the OSG has on top of Globus, the latest of which is gLExec, which is used for pilot [*technology from gLite*] jobs. Several of the big virtual organizations have this technology. You might have one guy sitting at FermiLab sending out Condor Glideins all the way across the Grid, and pilot uses gLExec to determine the appropriate userid the jobs should run under. In the big picture gLExec pushes responsibility for authentication and authorization to every single compute node. The software stack to support this is very complicated.

- Version consistency, standardization – that's clearly the name of the game here. The pace of change of some of the software is dizzying.

- What I'm looking for is like, when I install my favorite Linux distro (or cygwin on windows, or fink on the mac), I can go and pick out what I want, and it almost always works. You get a menu, you pick, it installs it, and everything works.

3.3.6 Other Issues

Several issues raised during the interviews did not fit into any other category.

3.3.6.1 Allocations

Some concerns focused on difficulties related to **allocations**:

- One of the biggest challenges is getting a large enough amount of computer time to do the calculations that we would really like to be able to do but just can't accomplish right now.

- The usual funding issues where you're writing grants and waiting six months to find out if you did or didn't get it – especially for solicitations that have less than a ten percent success rate – that is pretty counterproductive.

3.3.6.2 Community and collaboration

Other concerns focused on **community and collaboration-related issues**:

- One challenge is that there is so much development happening now, in so many directions, by so many people, that it's becoming harder and harder to keep track of all the developments. Trying not to reinvent things and trying to leverage what's already there is a constant challenge.

- People have a tendency to write and rewrite monitoring applications as if forgetting all the enormous amount of work done on this topic by people before them.

- For us, to do the research we want to do, we need to have Globus and (ideally) Condor and some other software using the new format logs. We also need to get OSG to deploy the central log file collection stuff. Some of these things are not super-high-priority items for everyone involved. So it can take a lot of phone calls and prodding. Everybody agrees it's a good idea. It's just not always on the top of everybody's priority stack.

- Generally speaking, when you need modifications or improvements to software you depend on, it’s not under your control. In most cases you do not directly control the developer resources to get those changes done. And so you have to go back and ask them, and you don't really have much to offer in return other than the greater good of what you're trying to do. We’ve been doing well operating under these constraints, but it hasn’t been easy all of the time. There have been times when we’ve been told no, flat-out, “No, we aren’t going to do it; I don’t care how much you need it.” And then other times when we’ve been told, “Yeah, but it will take awhile.” And in some cases, it takes a long while. Also there are other times when you get it right away. No one’s out to get you, but they all have their own agendas. Your requests need to be fit into larger priorities, and sometimes the requests are not given as high a priority as you would like. It’s not that they're trying to hurt you; it’s just that they have other bells to answer. That’s hard. There’s no way around it. I don’t know how else to put it. It can be a problem at times.

3.4 *Excerpts of Reported User Satisfaction*

Figure 5 shows an overview of the reports of user satisfaction in the study to date. Individual observations follow the figure.

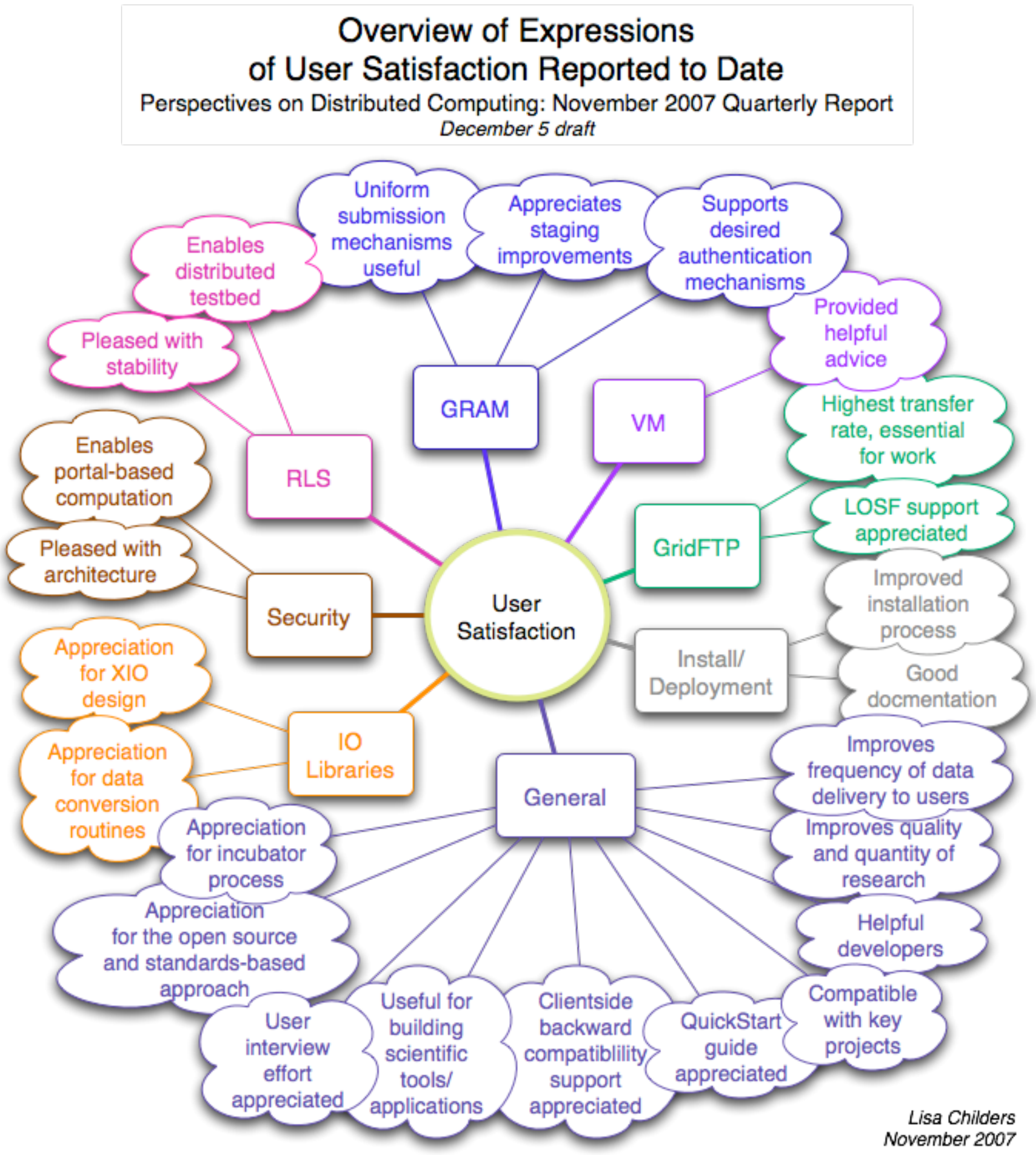

Overview of Expressions
of User Satisfaction Reported to Date
Perspectives on Distributed Computing: November 2007 Quarterly Report
December 5 draft
User Satisfaction
GRAM
Uniform submission mechanisms useful
Appreciates staging improvements
Supports desired authentication mechanisms
VM
Provided helpful advice
GridFTP
Highest transfer rate, essential for work
LOSF support appreciated
Install/ Deployment
Improved installation process
Good docmentation
General
Improves frequency of data delivery to users
Improves quality and quantity of research
Helpful developers
Compatible with key projects
QuickStart guide appreciated
Clientside backward compatiblility support appreciated
Useful for building scientific tools/ applications
User interview effort appreciated
Appreciation for the open source and standards-based approach
Appreciation for incubator process
IO Libraries
Appreciation for XIO design
Appreciation for data conversion routines
Security
Enables portal-based computation
Pleased with architecture
RLS
Enables distributed testbed
Pleased with stability
Lisa Childers
November 2007

Figure 5: Expressions of user satisfaction reported to date

3.4.1 General

- We went to a scientist who was highly computationally bound working on a small set of machines and created a portal environment to encapsulate his workflow. In the two week demo the scientist had access to far more compute cycles than he had been able accumulate to date. He said he got publishable work out of it. … The scientist has since gone on to get research funding to buy clusters and then contribute those clusters to our project.

- Globus is certainly the dominant technology. It is compatible with a lot of the larger projects we want to interact with.

- Client side backward compatibility is important. When a new version comes out, I should not have to rewrite my software. This has been a major concern in the past but has been much better lately.

- Globus rules! It's going to save the world! I'm not kidding. We rely quite heavily on Globus. Globus does all process management, the start-up, the security. In Globus we're using IO for all intermachine communication.

- I am attracted to the WSRF framework. I think that, especially for someone like myself who's not a computer science person, it allows me to quickly and easily leverage things like resource properties and the publication of resource properties. The lifetime management, the subscription and notification – all the nice things. It allows me to leverage them quickly and much more easily than I could do if I had to do all that stuff by myself. After all, I am a physicist and I'm dangerous when I'm writing code. The more code other people write for me, the better.

- I certainly appreciate all their efforts. The Globus Toolkit has really succeeded in what I think is one of its primary missions: enabling more science. Without a doubt, Globus has made more science possible. Period.

- I really appreciate the overall effort. For Grid the whole paradigm can only thrive is if there's an open source and standards-based implementation, and the Globus Toolkit is delivering exactly that. See one problem in the medical domain is that the internals of every equipment vendor, both software and hardware are proprietary. There are some standards on the interface side, but internally it's all proprietary. I think that the whole concept of service-oriented architecture presented in the Globus Toolkit and the Grid paradigm can have a major impact on how medicine is being addressed from a technology side.

- I want to say that I appreciate having people who build the software actually look at how people are using them. So this interview process is useful.

- I'd say within the last six to ten months we've been seeing more Globus developers showing up on some of the OSG calls and being there as a resource, if needed. I know where to find them if I need them.

- It's really not a lot of work to customize the Globus tools to fit their own users' use cases. I see TeraGrid doing that, and I think that's great.

- The Grid has made a qualitative difference in what we can do. For example, to analyze the data in preparation for a new release of our data products: If you want to do it on the cluster sometimes it's very difficult to get nodes. And on forty nodes the analysis will run for weeks. But on the Grid we can immediately do it and it will take so much less time. We can provide our users with fresh data more frequently because of the Grid.

- Keep doing it. The work is incredibly valuable, and I just don't think we're at a point where we can stop. It's like we're building a car and we haven't put the engine in, so we can't start it yet. When the time is right we're certainly going to start utilizing those services and those applications. Exactly how and when and other details have yet to be decided. But yeah, go for it.

- Thanks. But don't stop.

- Some people are perfectly ready for Grid technologies – for access to highly distributed computational resources. Those who say, "I've been waiting for you to come along" – they have an application that needs cycles or needs to move data or somehow needs a cluster. Dealing with these type of folks is so easy. We enjoy it so much. We give them credentials and adapt their application. Ours is Web services only, so we wrap it in Web services submission script. Maybe we even build them a small service. We drop the tech in and they're happy.

- The formalization of Web services I think is an incredible technology to allow machines to communicate with machines in a very standard and structured way, though there are still questions on what to use in the Web services area: Do I use SOAP? Do I use REST? Honestly, those types of remote procedure calls have been around for a long time in the Unix environment using sockets, but they were unique to those platforms. I think the convergence now with the standards is really making things fly.

- There's no other technology out there that provides compute resources, data management resources and security at the level that the Globus Toolkit does. Period.

- We really like Web services because it lets us build tools that are better suited to scientific workflows. They really are services – the steps that we need to accomplish.

- We’re working in the field of radiation modeling for cancer therapy, and there’s a proton accelerator here in the state that has been built by the M. D. Anderson Cancer Center. This large-scale $120 million facility has huge modeling needs. We thought it would be a very hard problem to move the medical data around. But we found that there are tools in the caGrid software stack that are not only well suited, they’re actually explicitly written for the purpose of moving medical image data around with high security using Grid tools. So that is an area where we thought we’d need to do a lot of development. Instead we found a complete working infrastructure that we just didn't know about.

- We’ve recently become an incubator project for our netlogger work. There have been no hassles other than trying to convince LBL lawyers that Apache and free BSD licenses are effectively the same thing – which is LBL's problem, not Globus's problem. I've been pretty impressed with the whole incubator process. I like the fact that you get a Wiki, a bug tracker, and a CVS repository. And if you need something configured it seems to happen pretty quickly. The lead *dev.globus* infrastructure person seems really good. I was impressed with the whole incubator startup process and how smoothly it all went.

- When computer scientists actually try to use existing software to do the tasks they’re writing new software for, they develop software that is highly domain-relevant. I don’t know whether acquiring this domain knowledge is something that needs to be done by the team itself, or whether they need to have close collaborations. I think the collaboration that we have with the University of Chicago Grid experts may be very valuable in that respect. As neuroscientists we've had our frustrations, and those frustrations are being solved by some of these new approaches. Actually, that’s not fair to say. They’re not being solved, but we’re working towards solutions. We'll see in five or eight or ten years whether we've really had a good effect. So I agree with the University of Chicago Grid experts’ approach, which is to be highly collaborative with the domain specialists. I applaud that and think it's the right way to go.

- With the Globus team I generally feel like when I ask questions they’re quite responsive.

- Based on our profession interactions with other large-scale centers, we can quite often implement a framework that would have been beyond the reach of a researcher left to his own devices. That is very satisfying. It isn’t really a Grid topic per se, but it intersects a lot with the whole idea of shared resources.

- Leveraging large software projects like MPICH and Globus (and even to a lesser degree GridFTP or UDP or UDT) is great because it’s a tremendous leg up. You leverage it. There’s a bunch of code that’s there, and you apply a little bit of work, and you get a tremendous benefit from it without having to do all of the work.

- Much of the basic functionality, such as data transport, is already included in the Globus Toolkit. And there was not a lot of effort required to make Grid technology work for the medical domain. And that was very neat because you don't have to reinvent the wheel.

- Thank you so much because somehow the use of the grid and the use of the Globus really, really, really made such a huge difference in what we are doing. We can do so much more science after using the grid. So just my deepest, deepest, deepest and sincere thank you.

- The Quickguide to installing the container is very good, very straightforward and clear. Even for somebody who is a beginner, this is a straightforward document.

- The toolkit model fits me exceptionally well. I'm really looking for tools and infrastructure that allow me to build on top of them. So they have to be extensible, they have to have nice APIs and hooks that I can layer my own stuff on top.

- We chose Globus because it seemed to be the dominant technology for Grid services. And we were interested in going the Web services direction. We haven't found a reason to revisit that decision.

- With Grid technology we deliver images from any clinical trial center into the radiologist's own review workstation. That's capable now with Grid technology. And that's a big reward for us to see that really happening. And the radiologists are very pleased with this. This actually engages more radiologists in clinical trials than before. So we can actually improve the quality and quantity of research being done.

3.4.2 GRAM

- GRAM is great. In one fell swoop it allows me to specify jobs in a single language, and hides all the details of every local job manager (which nobody wants to learn.) Plus, it has the entire security infrastructure built-in. That's a hard, hard problem to solve.

- GRAM4: because the staging support is better. It's a pretty big improvement over GRAM2 in that sense, because you can do smarter staging (like the whole filelist). That maps much better to our Condor description files. And in general the architecture is better.

- The Rendezvous Service was written to provide all-to-all exchange of information in the Web services context. We were able to whittle it down to a reasonable API. And it was even a great exercise because I showed up with a need that the GRAM developers didn't foresee. We talked with them and nailed down the requirements, and it was done. This was about a year or two ago. It was great.

- The GRAM interface is really cool and provides us with one common interface for all the job managers. This is important for us since for our requirements we need to use multiple clusters and each has its different job managers. GRAM enables us to use the uniform mechanisms for both job submission and monitoring.

- GRAM supports the authentication mechanism that we like.

- We didn't see any particular need to support pre-Web services. And so as an experiment we tried just GRAM4. It seemed to have advantages in terms of being stateful and allowing us to interact more closely with our potential services. Certainly from the point of view of just job submission it was relatively trivial to adopt.

- We use GRAM4 for our job submission. That's where we started two years ago. We've had a little bit of struggle along the way – of course, it's been improved. We really have never regretted that decision. We will support GRAM2 job submissions to our batch-oriented resources on request, but we've never had a request that couldn't be satisfied by teaching the person how to submit via GRAM4.

3.4.3 GridFTP

- Our major challenge has been the LSOF, but it looks to be addressed, and we're very excited about leveraging that functionality. Other than that … it's completely reliable and moves tons of data. What else could I want?

- It is the way that exposes the highest rate of network transfer from filesystem to filesystem across a transcontinental distance. So for example if I have to move output from Pittsburgh to San Diego, GridFTP buys me a factor of two or three over bbFTP, which I could look after myself. And that two - three is essential.

3.4.4 Installation and Deployment

- The Globus installation has improved a lot.

- I've been installing Globus from the first version to today. The installation is very good, the documentation – the problems we encounter during the installation – they've been documented very, very well.

3.4.5 I/O Libraries

- IO was very good, but XIO is even better because it allows us to build protocol stacks. The protocol stack design will make it much, much easier for us to introduce new technologies in the future.

- So, when it was all Globus IO-based we had to go through the exercise of pushing GridFTP into MPICH-G2. It didn't kill us, but it wasn't easy. We had to munge the code pretty heavily to shove that stuff in. And we had to go through the same exercise when we had to push UDT into MPICH-G2. With Globus XIO, all of this can be very neatly wrapped into an XIO module that either I or someone else writes. That's the recipe for rapid prototyping. I won't have to mess with the MPICH-G2 code at all anymore. I'll just write an XIO module, and one line of code to activate the module in MPICH-G2, and it's done. And you get it for free. That's a big step forward for us. That's a big help.

- The Globus data conversion library is indispensable. We need it. If it were to go away, we'd have to write it from scratch ourselves. MPICH-G2 is responsible for doing the data conversion between big endian and little endian machines, for example. The MPI application is not going to give a darn about that. I need to care about that. It's an ugly job that no application should have to write. One library should write it once and then provide it. We provide it in MPICH-G2 because the Globus developers wrote it.

3.4.6 RLS

- When RLS goes down, you better believe we know it. And we have to jump into action. Fortunately it very rarely goes down now. We're quite pleased.

- Our testbed can be distributed across multiple locations because of the replicas. The replicas also allow us to choose the fastest available compute server for our computations

3.4.7 Security

- The Grid security infrastructure is one of the things we completely rely on and are building our tools on. For example all these remote job submissions and remote file transfers are completely based on GSI. So we absolutely love GSI-based authentication. This is something that has been really helpful and paved the way for the portal-based computation.

- We'd be lost without X.509 authentication. That just solves a whole raft of problems.

- With the Grid security model, you can do quite a better job electronically: you can do an audit, verify certificates, verify attributes, etc. These mechanisms are way better than what clinical practice is right now, because many of the documents today are in writing, stored in physical files, reports end up in the trash, etc. So there are many places in the current system where private information is exposed to the outside. This is one way that I think Grid technology can help, because it has a very good security model.

- You just walk into a sysadmin's office these days and say, "We're using GSI - the Globus security infrastructure," and they get it. They know that it's been looked at by many eyes (maybe even their own). They trust it, so you don't have to convince them it is secure.

- If you have an IGTF-accredited CA, that's enough, because other large-scale projects throughout the world get these sets of trust anchors. So they know whether or not to trust the credentials of your CA, and on what basis. They know that you have, for example, been in-person identity-proofed by someone in the chain. They also know the CA is run in a method that does not allow a graduate student to walk in and issue their own certificates. So since these things are widely distributed and commonly accepted, it's very easy to start a virtual organization. We always make the point that authentication is not authorization, but it's a starting point. They can then know the quality of the Grid credentials coming in and use that as a basis for signing membership to the Virtual Organization. It becomes barrier-lowering because I can accept a certificate issued in Czechoslovakia, for instance.

- The Globus GSI and everything that's built around that ecosystem now works really well for us. We can hook into it in so many different ways. We can set up services that manage the delegation for the users. The only thing the users have to do is enter the system once using something like MyProxy. Everything else is handled for them. That works really well.

3.4.8 Virtual Machines

- For redundancy we're setting up two physical machines, each with four virtual servers. We're using Xen to do the virtual server. We talked to the leader of the Globus Virtual Workspaces service and received some advice in the early stages that helped us figure out what to do there. It was very helpful.

4 Interim Recommendations

This section contains ideas for improving the scientific technology user's experience. The recommendations are inspired by the user interviews. First, a couple of disclaimers: (1) this is an interim report – recommendations are not final, and (2) the work outlined here will be weighed independently against development priorities outside the scope of the User Perspectives project.

Individual bugzilla entries have been created to represent each recommendation so that progress and decisions regarding each issue can be tracked. As is the case with all CDIGS planning documents, members of the community are welcome to make their views known via bugzilla.

Recommendations are listed in no particular order.

4.1 Error Codes

Featured quote: *"Another type of information that is lacking is documentation about errors. For example with GRAM, all we get is an error code and there is not enough documentation explaining the error. We then have to google to find out how other users handled the problem. Sometimes we even need to go as far as to dig into the GRAM source code to determine under what conditions the error code is sent. There is some documentation, but not at a level enough that we can use it."* – User2

Recommendation #1: Document all Globus error codes
Identify and document all error codes currently issued by each Globus service, annotating each entry with advice that helps the user move past the error.

To participate in discussion about this recommendation please visit
http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5700

Project names of the users reporting relevant issues: *LEAD, ENZO, Lattice QCD, Engage VO*

4.2 Resource Properties

Featured quote: *"Solving problems is easy once you have all the data in front of you. It's getting the data and knowing what data to get that's the hard part. Networks are notorious for this, right? They're black boxes. Very rarely are you lucky enough to have access to somebody who can actually find out operational status on routers and the like. So you have to infer what's happening by using things like Iperf, netperf, pipechar, etc."*
– User10

Recommendation #2: Add useful resource properties
Leverage the WSRF infrastructure, populating services with information for diagnosing problems remotely. Include different types of information targeted at various users (service developers, sysadmins, client users, etc.). Distribute user-friendly accessors for getting at the diagnostic data.

To participate in discussion about this recommendation please visit
http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5701

Project names of the users reporting relevant issues: *LEAD, Lattice QCD, TIGRE, ALCF, Engage VO, FermiGrid*

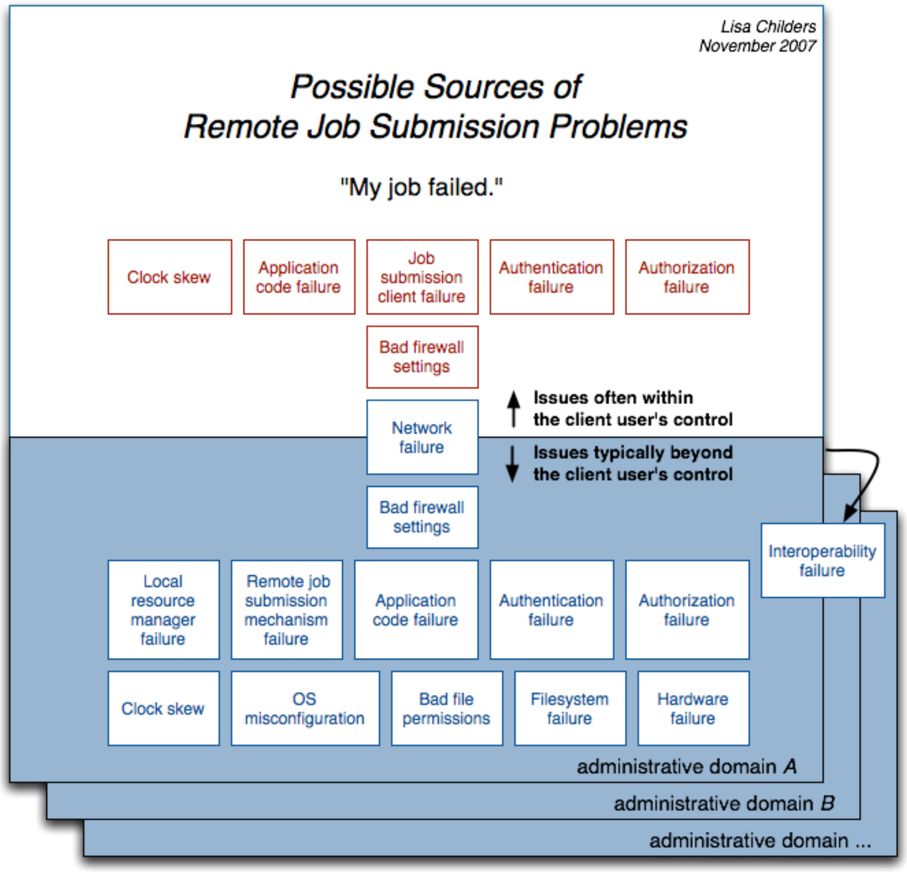


Figure 6 Service reliability – a nontrivial issue

4.3 Failure

Featured quote: *"I'm running in this 2,000-4,000 CPU range at the moment. And within the next year we expect that to go up to at least the 32,000 CPU range, if not a factor of two more than that. The unreliability that I see in filesystems, even in batch-process launching systems, disks, monitoring tools – you name it – nothing really works reliably at the 2,000- or 4,000-processor level today. I am extremely doubtful about it working at a level ten times greater than that."* – User4

Recommendation #3: Make failure handling a high-priority issue for CDIGS development

The topics of reliability and failure handling are becoming increasingly critical as state-of-the-art cyberinfrastructure moves into production. Figure 6 illustrates many possible sources of failure that can disrupt a user job. Only those technologies demonstrating resiliency in the face of the various types of failures will succeed.

To participate in discussion about this recommendation please visit http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5702

Project names of the users reporting relevant issues: *LEAD, ENZO, TIGRE, Lattice QCD, LIGO, FermiGrid*

4.4 Data Movement

Featured quote: *"The tools that we use to move files typically are the standard unix tools included with ssh. And for that we don't need more information it's just painful. Painful in the sense of having to manage the transfers by hand, restarting transfers when they fail - all of this is done by hand."* – User6

Recommendation #4: Perform a market analysis of data movement patterns and use cases in the field today
A subset of scientists interviewed describes problems moving data around the country despite the existence today of useful tools designed to help alleviate the problems. The recommended analysis should provide the information needed to get Globus data services deployed and properly configured on the right machines in order to help these people.

To participate in discussion about this recommendation please visit
http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5703

Project names of the users reporting relevant issues: *ENZO, Lattice QCD, PUMA*

4.5 Educational Materials

Featured quote: *"Once I do find the documentation it's very hard to understand - it's full of acronyms, and refers to unfamiliar and unnatural concepts."* – User8

Recommendation #5: Create educational material written to the user's point of view (See Appendix B for example slides)

To participate in discussion about this recommendation please visit
http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5704

Project names of the users reporting relevant issues: *LEAD, Lattice QCD, ALCF, PASTA*

Appendix A: Overview of the User Advocacy Program

Figure 7 summarizes current near-term goals for User Advocacy. The remainder of Appendix A discusses the motivation for the User Advocacy program, outlines additional goals, and lists the accomplishments to date.

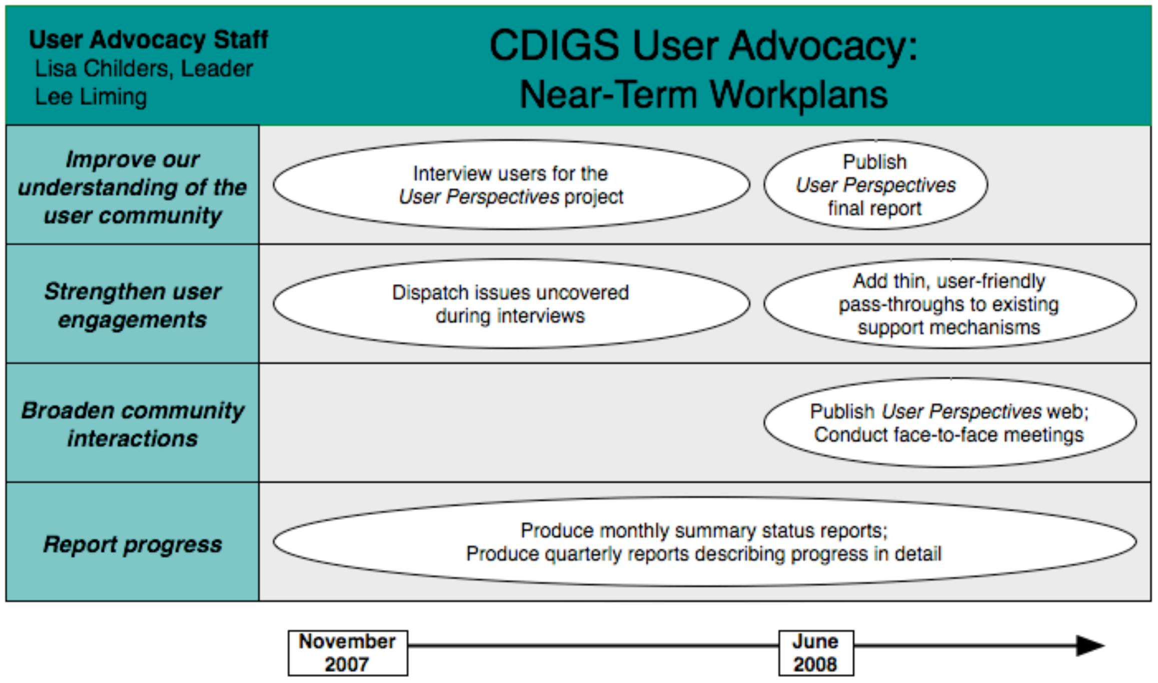

User Advocacy Staff
Lisa Childers, Leader
Lee Liming
CDIGS User Advocacy: Near-Term Workplans
Improve our understanding of the user community
Interview users for the User Perspectives project
Publish User Perspectives final report
Strengthen user engagements
Dispatch issues uncovered during interviews
Add thin, user-friendly pass-throughs to existing support mechanisms
Broaden community interactions
Publish User Perspectives web; Conduct face-to-face meetings
Report progress
Produce monthly summary status reports; Produce quarterly reports describing progress in detail
November 2007
June 2008

Figure 7: Current User Advocacy workplan

A.1 Background

A.1.1 Motivation

Users of scientific distributed computing technology face many challenges today. Harried middleware developers are pressured to deliver and support lightly tested, sparsely documented features. Users are pressured to quickly get things running in order to accomplish their own work. Everyone is busy, and the software systems being built are as complex as they are powerful.

Given these circumstances, users sometimes have trouble efficiently getting tech support, in part because it is not always clear how their issues map onto existing tech support systems. While the *dev.globus* community forum is a powerful and rich tool for interacting with Globus development, it is implementation-focused. Though it is appropriate for developers, those unfamiliar with Globus implementation details might find it daunting.

The overriding purpose of the User Advocacy effort is to provide the CDIGS team with additional information about current and potential users so its work can be made more accessible to scientific users. Armed with information provided by the User Advocacy effort, the CDIGS team will be more effective in tuning its tech support interfaces, educational materials, and development priorities.

A.1.2 Relation to Current Work

Outreach
Outreach is an activity focused on disseminating information about the group's current and future work. The Outreach team is responsible for informing the community about Globus technologies and development activities and generally advocating for "The Globus Way." There is no question that the current Outreach work is essential to our success as a group, but it has a different focus from that of the User Advocacy program. User Advocacy is user-centric. The overriding goal of the User Advocacy effort is to better understand and represent the users' view within the team, without allegiance to any particular technology. As such, User Advocacy seeks to complement the Outreach effort.

Project Coordination
We do have user advocacy-like functions in the group right now, such as the targeted user engagements with ESG. But given our CDIGS mandate, we need to extend the reach of these activities. The traditional Globus project coordination model of assigning a representative to each project will not scale to the degree needed. Over time the User Advocacy program will represent the broad interests of a variety of scientific projects. As such, User Advocacy seeks to augment current project advocacy efforts.

Community Forum
Navigating the technology-centric organization of our current support infrastructure can be confusing for the naïve. User Advocacy deliverables in the customer support area will involve adding thin, user-friendly pass-throughs into existing support mechanisms. The exact design and approach for this work are not yet known. However, it is safe to say that User Advocacy seeks not to replace or duplicate existing support mechanisms but to provide new paths into them.

A.1.3 Staffing

The CDIGS User Advocacy program is led by Lisa Childers, with Lee Liming participating as available.

A.2 User Advocacy Workplan

The User Advocacy workplan has four major goals, each discussed below.

A.2.1 Improve Our Understanding of the User Community

Before any real change can take place, User Advocacy staff must investigate the concerns and problems facing users today. The cornerstone of this investigation is the recently initiated User Perspectives project.

The User Perspectives project will produce a view of users' goals, methods and problems, collected and documented in a methodical and rigorous fashion. Key deliverables include the following:

- Early draft of the User Perspectives report framework (June 2007)
- Additional interviews each month (July 2007–April 2008). The objectives are threefold:
    - Identify individual issues requiring follow-up
    - Forge connections and create goodwill among current and potential users
    - Gather ideas for improving our software, tutorials, and documentation
- Creation of the Perspectives on Distributed Computing final report (June 2008)

A.2.2 Strengthen User Engagements

Over the coming year, the User Perspective interviews will also be used as an opportunity to deepen our engagement with a variety of users with whom we might otherwise not directly interact. Initial deliverables on this front include the following:

- Bug reports and follow-up on individual issues as needed
- Identification of new collaboration opportunities

A.2.3 Broaden Community Interactions

In time the User Advocacy effort will add a new dimension to Globus community interactions. User Advocacy staff will work to build a crosscutting, user-focused community. Deliverables include the following:

- User Perspectives companion Web
- Face-to-face meetings (to be determined)

A.2.4 Report Progress

Interim reports will be generated to provide a view on the project's progress:

- Monthly CDIGS status reports
- Monthly bundles of raw interview data
- Quarterly reports summarizing interim findings of the User Perspectives project and User Advocacy accomplishments (August 2007, November 2007, March 2008, June 2008)

A.3 User Advocacy Accomplishments: May–November 2007

A.3.1 Improve Our Understanding of the User Community

- Founded the User Perspectives project (Childers, May 2007)
    - Designed interview questions (Childers and Liming, May 2007)
    - Conducted test interview (Childers, May 2007)
    - Designed the User Perspectives report framework (Childers, May–June 2007)
    - Reviewed approach with HCI expert (Liming, June 2007)
- Conducted thirteen user interviews (Childers, May–July 2007)
    - Observed five user interviews (Liming, May–July 2007)
    - Transcribed nine user interviews, totaling approximately 13 interview hours and 47,000 user words (Childers, May–July 2007)

- Wrote Perspectives on Distributed Computing: August 2007 Quarterly Report (Childers & Liming, August 2007)
- Conducted eighteen user interviews (Childers, August–October 2007)
    - Observed one user interview (Liming, September 2007)
    - Transcribed ten user interviews, totaling approximately eleven interview hours and 42,000 user words (Childers, August–October 2007)
- Applied for and received IRB approval (Childers and Foster, October 2007)
- Wrote Perspectives on Distributed Computing: November 2007 Quarterly Report (Childers and Liming, November 2007)

A.3.2 Strengthen User Engagements

- Sent follow-up email to QCD scientist requesting more information on GridFTP transfer problem (Childers, June 2007)
- Dispatched sixteen user issues to CDIGS staff members for near-term follow-up (Childers, August 2007)
- Dispatched two user issues to CDIGS staff for near-term follow-up (Liming, November 2007)
- Formulated five recommendations for the CDIGS team, designed to improve the scientific user's experience with Globus software (Childers and Liming, November 2007)

A.3.3 Broaden Community Interactions

No significant accomplishments during the reporting period

A.3.4 Report Progress

- May 21, 2007: Transcription of demonstration interview sent to Fraser, Foster, Liming
- June 15, 2007: CDIGS monthly status reported
- June 18, 2007: Childers, Foster, Fraser, and Liming reviewed User Advocacy proposal and three interview transcriptions
- July 2, 2007: Five interview transcriptions sent to team members
- August 18, 2007: CDIGS monthly status reported
- August 28, 2007: Slides created for 2007 CDIGS NSF review
- August 29, 2007: First nine interview transcriptions sent to team members
- August 29, 2007: Quarterly report sent to CDIGS team
    - *This report was called out for commendation by the NSF CDIGS review panel in September 2007*
- October 15, 2007: CDIGS monthly status reported
- October 17, 2007: Four interview transcriptions sent to Liming
- November 15, 2007: CDIGS monthly status reported
- November 19, 2007: Four interview transcriptions sent to Liming
- December 2, 2007: Material provided for CDIGS annual report
- December 6, 2007: Quarterly report sent to CDIGS team
- December 6, 2007: Ten interview transcriptions sent to team members

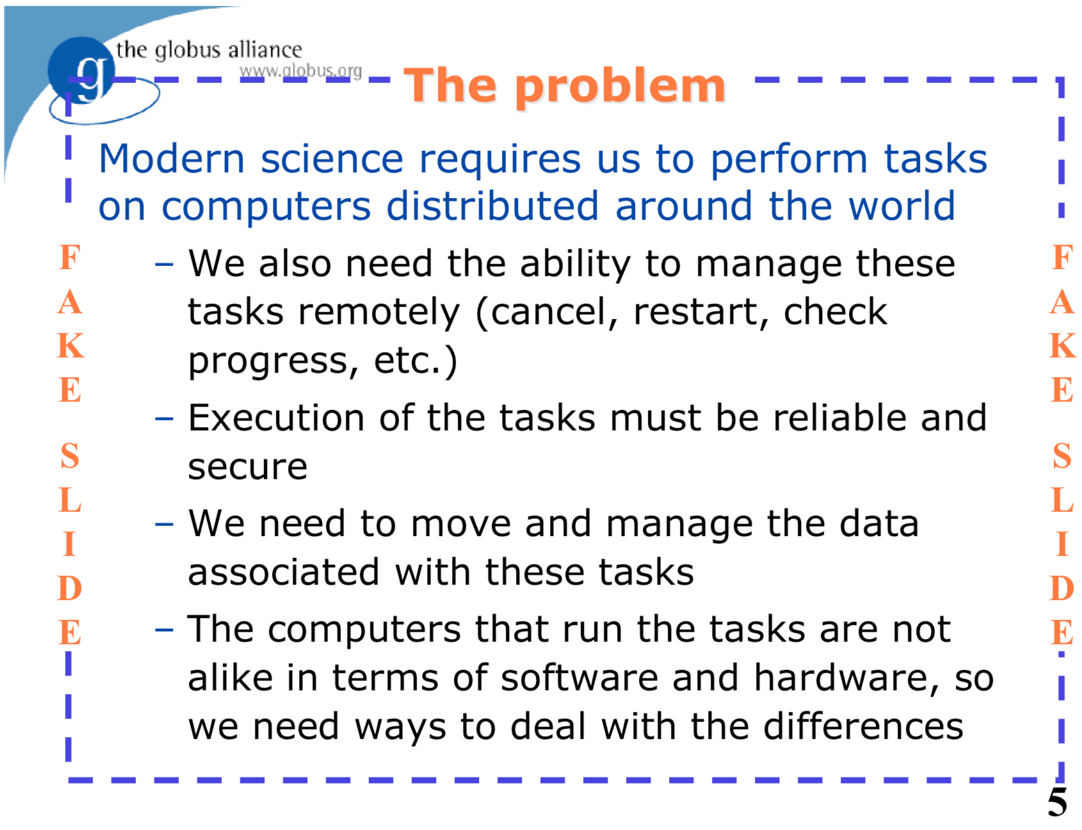
Appendix B: User-Focused Slide Example

Too often technology presentation materials focus on implementation details that are irrelevant and confusing to all but a few. What follows is an example set of slides illustrating one approach for introducing technological information to new users.

Note: these slides are for demonstration purposes only – this is not official Globus educational material.

The slides were written by using the following principles:

- Determine the appropriate public interface for describing the technology to the user
- Do not mention implementation details beneath the relevant public interface layer
    - For the scientist: "This Globus phone sends and receives text messages"
    - For the middleware developer: "This Globus phone is GSM-compatible"
- Prepare distinct explanations for the various user types
    - Scientist, application/portal developer, system administrator, etc.

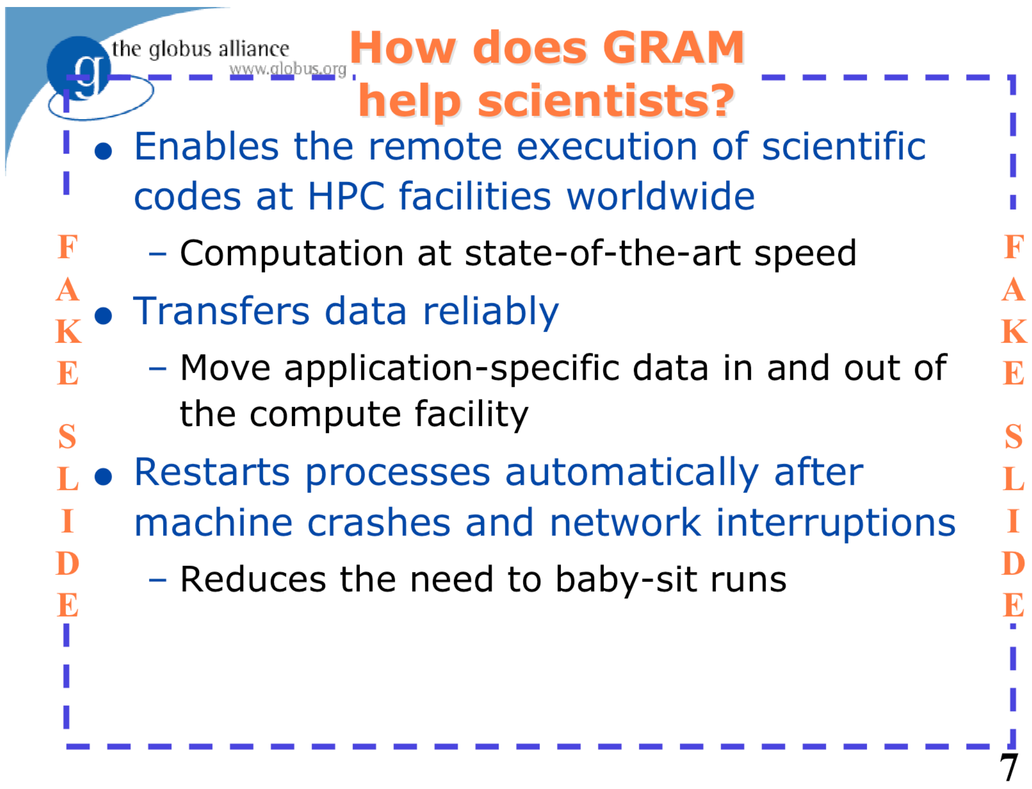

the globus alliance
www.globus.org
FAKE SLIDE
FAKE SLIDE

The problem

Modern science requires us to perform tasks on computers distributed around the world

- We also need the ability to manage these tasks remotely (cancel, restart, check progress, etc.)
- Execution of the tasks must be reliable and secure
- We need to move and manage the data associated with these tasks
- The computers that run the tasks are not alike in terms of software and hardware, so we need ways to deal with the differences

5


the globus alliance
www.globus.org

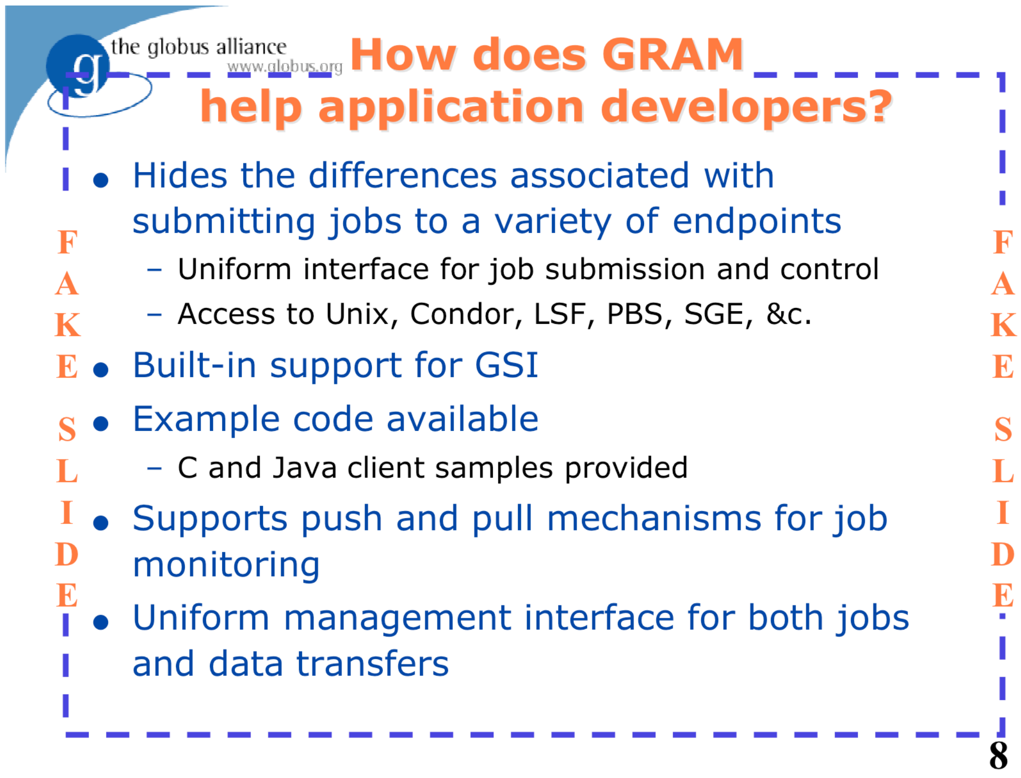
FAKE SLIDE

How does GRAM help scientists?

- Enables the remote execution of scientific codes at HPC facilities worldwide
  - Computation at state-of-the-art speed
- Transfers data reliably
  - Move application-specific data in and out of the compute facility
- Restarts processes automatically after machine crashes and network interruptions
  - Reduces the need to baby-sit runs

FAKE SLIDE

7


the globus alliance
www.globus.org

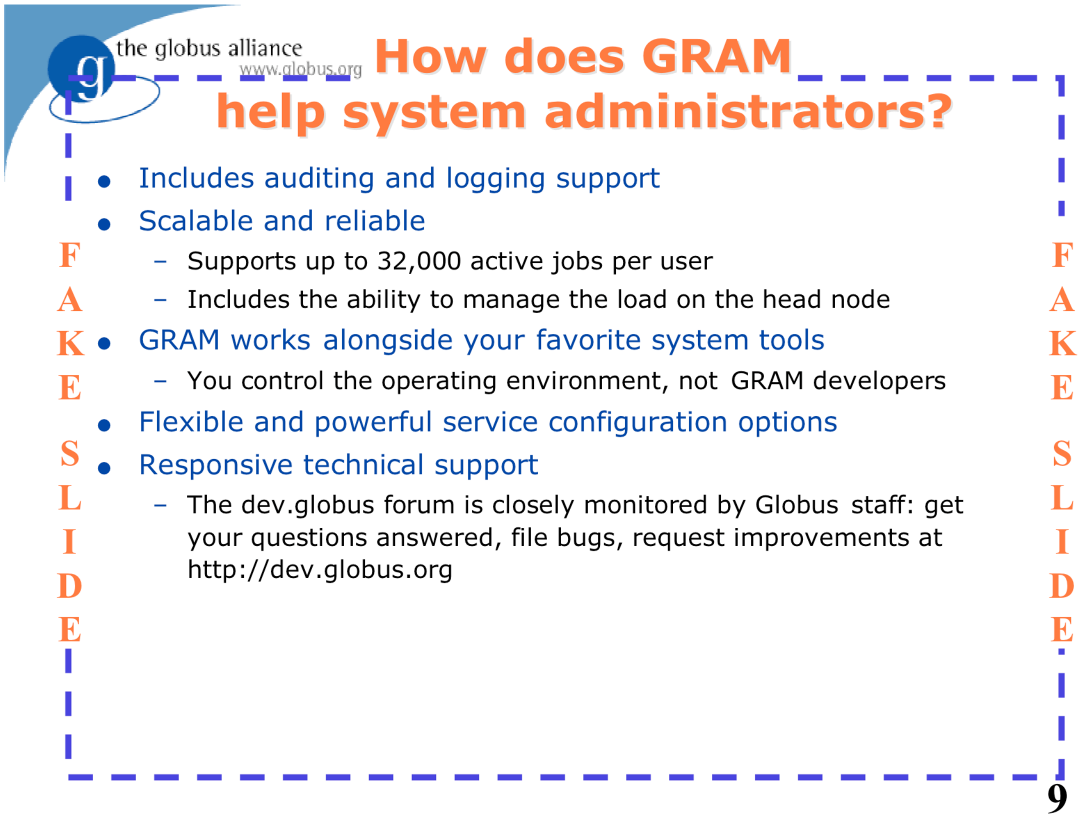
FAKE SLIDE

How does GRAM help application developers?

- Hides the differences associated with submitting jobs to a variety of endpoints
  - Uniform interface for job submission and control
  - Access to Unix, Condor, LSF, PBS, SGE, &c.
- Built-in support for GSI
- Example code available
  - C and Java client samples provided
- Supports push and pull mechanisms for job monitoring
- Uniform management interface for both jobs and data transfers

FAKE SLIDE

8



the globus alliance
www.globus.org

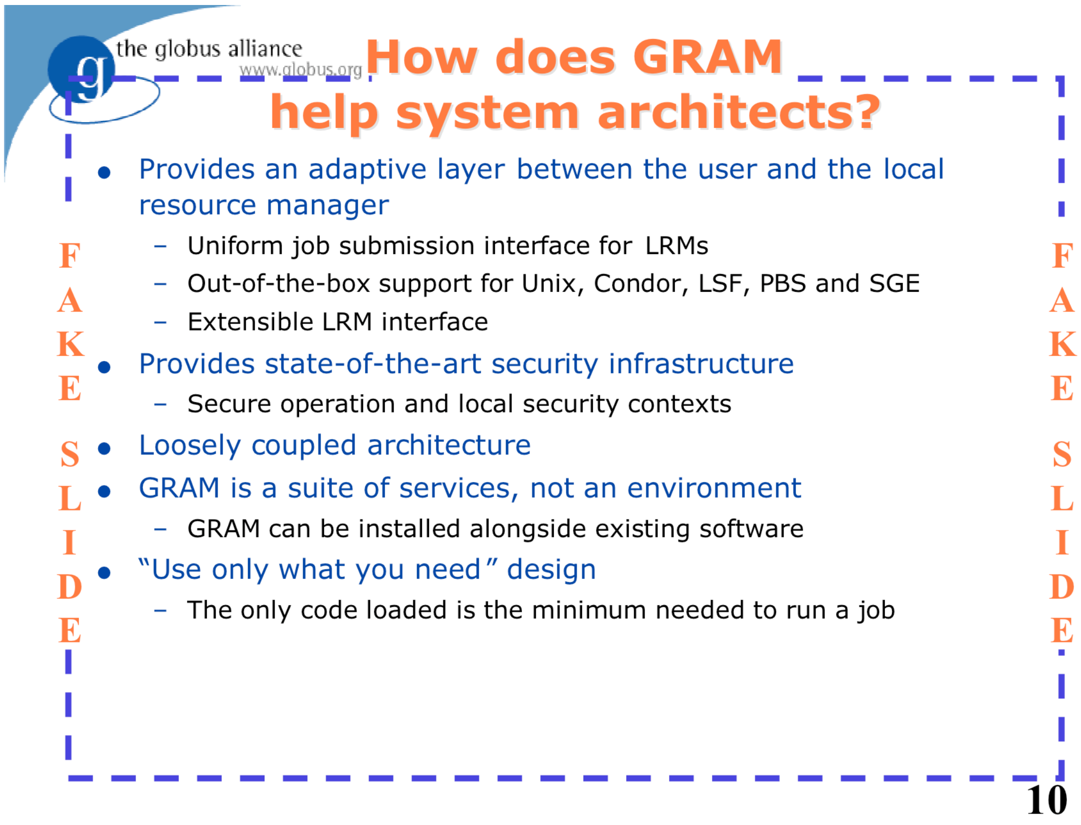
How does GRAM help system administrators?

FAKE SLIDE

- Includes auditing and logging support
- Scalable and reliable
  - Supports up to 32,000 active jobs per user
  - Includes the ability to manage the load on the head node
- GRAM works alongside your favorite system tools
  - You control the operating environment, not GRAM developers
- Flexible and powerful service configuration options
- Responsive technical support
  - The dev.globus forum is closely monitored by Globus staff: get your questions answered, file bugs, request improvements at http://dev.globus.org

FAKE SLIDE

9



the globus alliance
www.globus.org

How does GRAM help system architects?

FAKE SLIDE

- Provides an adaptive layer between the user and the local resource manager
  - Uniform job submission interface for LRMs
  - Out-of-the-box support for Unix, Condor, LSF, PBS and SGE
  - Extensible LRM interface
- Provides state-of-the-art security infrastructure
  - Secure operation and local security contexts
- Loosely coupled architecture
- GRAM is a suite of services, not an environment
  - GRAM can be installed alongside existing software
- "Use only what you need" design
  - The only code loaded is the minimum needed to run a job

FAKE SLIDE

10